

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,
д-р техн. наук, проф.

_____ І. М. Журавська

«__» _____ 2024 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

VPN-сервер на базі Raspberry PI

Спеціальність 123 Комп'ютерна інженерія

123 – КБР.01 – 405.22010523

Студент

_____ П. Р. Чередніченко
підпис

«__» _____ 202__ р.

Керівник ст. викладач

_____ І. С. Бурлаченко
підпис

«__» _____ 202__ р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

« _____ » _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної бакалаврської роботи

Видано студенту групи 405 факультету комп'ютерних наук

Чередніченко Павло Русланович

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

VPN-сервер на базі Raspberry Pi

Затверджена наказом по ЧНУ ім. Петра Могили від 30.01.2024 № 17.

2. Строк представлення кваліфікаційної роботи « _____ » _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Успішне розгортання VPN сервера на базі Raspberry Pi, який забезпечить
безпечний та стабільний віддалений доступ до локальної мережі. Для
реалізації проекту необхідні наступні початкові дані: обладнання, програмне
забезпечення користувацькі дані конфігураційні
дані.

4. Перелік питань, що підлягають розробці

- 1) Аналітичний огляд програмного забезпечення конфігурації VPN-серверів та алгоритми керування розумними будинками;
- 2) Огляд і аналіз існуючих технологій рішень розгортання VPN-серверів;
- 3) Визначення технічних вимог та постановка задачі для розробки програмного забезпечення для різних сценаріїв мережевої безпеки розумних будинків;
- 4) Моделювання та технічне проектування методи шифрування протоколів обміну даними для різних режимів роботи VPN-серверів;
- 5) Програмна реалізація вебзастосунку для VPN-серверу;

5. Перелік графічних матеріалів

Схема підключення пристрою

Блок-схема алгоритму налаштування VPN-серверу

Блок-схема мостування VPN-серверу

6. Завдання до спеціальної частини

Провести оцінку умов праці на робочих місцях за основними факторами: температура, вологість, освітленість, рівень шуму, інтенсивність теплового випромінювання, концентрація шкідливих речовин та пилу. Визначити можливі ризики та негативні фактори, які можуть впливати на працівників під час роботи. Розробити конкретні заходи з покращення умов праці, включаючи технічні та організаційні рішення.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Макарова О.В. ст. викл.	кафедра екології Медичного інституту ЧНУ імені Петра Могили	Спеціальна частина з охорони праці

Керівник роботи

ст. викладач кафедри комп'ютерної інженерії Бурлаченко Іван Сергійович
(посада, прізвище, ім'я, по батькові)

(підпис)

Завдання прийнято до виконання

Чередніченко Павло Русланович

(прізвище, ім'я, по батькові студента)

(підпис)

Дата видачі завдання « ____ » _____ 20 ____ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: VPN-сервер на базі Raspberry PI

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КР	11.12.23	12.12.23	Виконано
2	Огляд літератури за темою роботи	15.01.24	18.02.24	Виконано
3	Складання календарного плану БКР	19.02.24	06.03.24	Виконано
4	Аналіз предметної області	19.02.24	04.03.24	Виконано
5	Розробка проектних рішень	23.02.24	09.03.24	Виконано
6	Моделювання та конструювання АПЗ	20.02.24	27.02.24	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	01.03.24	04.03.24	Виконано
8	Відгук керівника КР	09.03.24	07.04.24	Виконано
9	Оформлення БКР та презентації	15.03.24	21.04.24	Виконано
10	Попередній захист	13.06.24	13.06.24	Виконано
11	Рецензування	12.05.24	20.05.24	Виконано
12	Завершення оформлення КР та презентації	30.05.24	30.05.24	Виконано
13	Захист бакалаврської кваліфікаційної роботи	25.06.24	25.06.24	Виконано

Розробив здобувач ВО Чередніченко Павло Русланович _____
(прізвище, ім'я, по батькові) (підпис)
« ____ » _____ 20__ р.

Керівник роботи ст. викл. Бурлаченко Іван Сергійович _____
(посада, прізвище, ім'я, по батькові) (підпис)
« ____ » _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«VPN-сервер на базі Raspberry Pi»

Студент 405 гр.: Чередніченко Павло Русланович

Керівник: ст. викладач Бурлаченко Іван Сергійович

Тема кваліфікаційної роботи "VPN сервер на базі Raspberry Pi" присвячена дослідженню та впровадженню ефективного та безпечного рішення для віддаленого доступу до мережі домашньої панелі керування розумного будинку. Основна мета роботи полягає у створенні VPN сервера, що функціонує на основі мікрокомп'ютера Raspberry Pi, забезпечуючи безперешкодний і захищений доступ до вебзастосунку home automation dashboard, який слугує для управління розумним будинком. Використання VPN дозволяє мінімізувати ризики несанкціонованого доступу та захистити персональні дані.

Система включає мікрокомп'ютер Raspberry Pi, microSD карту для зберігання даних, USB кабель та блок живлення мережевий для Raspberry Pi.

Програмний код написано мовою програмування Python з використанням моніторингової системи Prometheus та сервісу Grafana. Код обробляє сигнали від сенсорів у розумному домі, передає у систему Prometheus і виводить значення у панель керування розумного дому.

У роботі детально розглядаються технічні аспекти налаштування VPN сервера, включаючи вибір програмного забезпечення, конфігурацію мережевих параметрів та забезпечення надійного захисту даних. Окрім того, досліджується інтеграція вебзастосунку home automation dashboard, який надає користувачам інтуїтивно зрозумілий інтерфейс для контролю різних систем розумного будинку.

Сторінок – 94. Рисунків – 51. Таблиць – 2. Посилань – 27. Додатків – 3.

Ключові слова: Raspberry Pi, VPN, розумний будинок, home automation dashboard, захист даних.

ABSTRACT

of the Bachelor's Thesis

"VPN server based on Raspberry Pi"

Student: Cherednichenko Pavlo Ruslanovich

Supervisor: Burlachenko Ivan Sergiyovich

The topic of Bachelor's Thesis "VPN server based on Raspberry Pi" is devoted to the research and implementation of an effective and secure solution for remote access to the network of the smart home control panel. The main goal of the work is to create a VPN server that operates on the basis of a Raspberry Pi microcomputer, providing unhindered and secure access to the home automation dashboard web application, which is used to control a smart home. Using a VPN minimizes the risk of unauthorized access and protects personal data.

The system includes a Raspberry Pi microcomputer, a microSD card for data storage, a USB cable and a power supply for the Raspberry Pi.

The software code is written in the Python programming language using the Prometheus monitoring system and the Grafana service. The code processes signals from the sensors in the smart home, transmits them to the Prometheus system, and displays the values in the smart home control panel.

The paper discusses in detail the technical aspects of setting up a VPN server, including software selection, configuration of network parameters, and ensuring reliable data protection. In addition, the integration of the home automation dashboard web application, which provides users with an intuitive interface for monitoring various smart home systems, is explored.

Pages – 94. Figures – 51. Tables – 2. References – 27. Appendices – 3.

Keywords: *Raspberry Pi, VPN, smart home, home automation dashboard, personal data protection.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....	3
ВСТУП	4
1 АНАЛІЗ РОЗРОБКИ VPN СЕРВЕРУ НА БАЗІ RASPBERRY PI.....	6
1.1 Актуальність теми.....	6
1.2 Огляд існуючих рішень системи, що розробляється	7
1.3 Формування вимог до апаратно-програмного забезпечення	13
1.4 Масштабованість та оптимізація продуктивності апаратно- програмного забезпечення	15
1.5 Технологічне рішення	16
Висновки до розділу 1	19
2 ПРОЄКТУВАННЯ ЗАХИЩЕНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ З ВИКОРИСТАННЯМ VPN СЕРВЕРІВ.....	20
2.1 Налаштування VPN сервера	20
2.2 Налаштування мережевого моста з OpenVPN	26
2.3 Криптографічні алгоритми шифрування.....	31
2.4 Підключення користувача до VPN серверу	36
Висновки до розділу 2	39
3 РОЗРОБЛЕННЯ АПАРАТНО-ПРОГРАМНОЇ ЧАСТИНИ ПАНЕЛІ КЕРУВАННЯ РОЗУМНИХ БУДИНКІВ	41
3.1 Налаштування VPN серверу на Raspberry PI	41
3.2 Веб-застосунок Home Automation Dashboard.....	45
3.3 Програмна реалізація.....	46
Висновки до розділу 3	66
ВИСНОВКИ.....	67
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	68
ДОДАТОК А Звіт з антиплагіату	71
ДОДАТОК Б Код отримання даних з сенсорів	72
ДОДАТОК В Код конфігурації панелей дашбордів.....	75

ПЕРЕЛІК СКОРОЧЕНЬ

VPN	– Virtual Private Network
RPI	– Raspberry Pi
HAD	– Home automation dashboard
ORM	– Object-Relational Mapping
AES	– Advanced Encryption Standard
DES	– Data Encryption Standard
CBC	– Cipher Block Chaining
GCM	– Galois/Counter Mode
ECDH	– Elliptic-Curve Diffie-Hellman
HTTP	– HyperText Transfer Protocol Secure
JSON	– JavaScript Object Notation
NAS	– Network Attached Storage
IP	– Internet Protocol
SSL	– Secure Socket Layer

ВСТУП

У сучасному світі технологій віддалений доступ та управління системами розумного будинку стали невід'ємною частиною повсякденного життя. Відповідно, зростає потреба у розробці безпечних і надійних рішень для забезпечення цього доступу. Метою моєї кваліфікаційної роботи є створення VPN сервера на базі RPI для захищеного керування домашньою автоматизацією за допомогою вебзастосунку HAD. Такий підхід дозволяє не лише спростити управління різними системами будинку, а й гарантувати безпеку та конфіденційність переданих даних.

Метою кваліфікаційної роботи бакалавра є підвищення безпеки вебзастосунків розумних будинків за рахунок автоматизації налаштування VPN-серверів на базу RPI

Об'єкт: керування автоматизацією розумних будинків.

Предмет: технології розгортання VPN-серверів на базі RPI.

Для виконання бакалаврської кваліфікаційної роботи треба виконати наступні завдання:

- проаналізувати програмне забезпечення конфігурації VPN-серверів та алгоритми керування розумними будинками;
- дослідити математичні методи шифрування протоколів обміну даними для різних режимів роботи VPN-серверів;
- спроектувати схему підключення пристроїв до VPN серверів;
- розробити програмне забезпечення для різних сценаріїв мережевої безпеки розумних будинків;
- розробити спеціальну частину з охорона праці;

Практичне значення отриманих результатів:

Даний VPN-сервер захищає дані від несанкціонованого доступу та можливих атак. Це особливо важливо для користувачів розумних домів, оскільки дозволяє приховати інформацію про те, коли і як вони використовують вебзастосунок HAD.

Після завершення етапу проектування, здійснюється безпосереднє створення апаратно-програмного комплексу, що включає налаштування операційної системи, встановлення та конфігурацію VPN сервера, а також інтеграцію вебзастосунку HAD. Завершальним етапом є тестування розробленого рішення, під час якого перевіряється його функціональність, стабільність роботи та рівень захисту даних. Результати тестування підтверджують ефективність створеного VPN сервера, що забезпечує надійний та безпечний доступ до систем розумного будинку, демонструючи високу наукову та практичну значущість виконаної роботи.

1 АНАЛІЗ РОЗРОБКИ VPN СЕРВЕРУ НА БАЗІ RASPBERRY PI

1.1 Актуальність теми

Предметна сфера цієї кваліфікаційної роботи охоплює кілька важливих аспектів сучасних інформаційних технологій, зокрема VPN технології, апаратно-програмні комплекси на базі мікрокомп'ютерів RPI та системи домашньої автоматизації. Застосування VPN дозволяє створити захищений канал для передачі даних через Інтернет, що є критично важливим для забезпечення безпеки конфіденційної інформації. Використання RPI, завдяки його доступності та можливостям, дозволяє створювати компактні та ефективні рішення для різних застосувань, включаючи домашню автоматизацію. Вебзастосунок HAD забезпечує зручний інтерфейс для керування системами розумного будинку, що дозволяє інтегрувати різні пристрої та забезпечити їхню злагоджену роботу[3].

У сучасному світі інтернету речей та розумних будинків питання безпеки даних набуває особливої важливості. Розумні будинки включають безліч пристроїв, таких як датчики, камери, системи освітлення та опалення, які можуть бути підключені до Інтернету та керовані дистанційно. Віддалений доступ до цих пристроїв забезпечує зручність і комфорт, але водночас створює потенційні загрози для безпеки. Неконтрольований доступ або атаки на такі системи можуть призвести до витоку конфіденційної інформації або навіть до фізичної шкоди. Тому впровадження VPN сервера, що забезпечує захищений канал зв'язку, є критично важливим для збереження конфіденційності та цілісності даних у системах домашньої автоматизації.

Розвиток технологій розумних будинків стає все більш популярним, оскільки вони надають значні переваги у зручності, ефективності та безпеці житлових приміщень. Вебзастосунки, такі як HAD, дозволяють користувачам інтегрувати та керувати різноманітними системами і пристроями з єдиного інтерфейсу. Ці застосунки допомагають оптимізувати споживання енергії, забезпечують дистанційний контроль і моніторинг, а також підвищують

загальний рівень комфорту. Проте, для забезпечення безпеки таких систем необхідні надійні рішення, що запобігають несанкціонованому доступу. Впровадження VPN серверів на базі доступних і потужних пристроїв, таких як RPI, дозволяє розробляти захищені та ефективні системи домашньої автоматизації, що робить тему дослідження надзвичайно актуальною у наш час[4].

У проєкті використовуються такі технології, як OpenVPN та Wireguard. Ці технології дозволяють досягти високого рівня безпеки та ефективності. Вибір між цими технологіями залежить від конкретних вимог та умов експлуатації, проте обидві вони є надійними інструментами для створення захищених каналів зв'язку у системах домашньої автоматизації. Таким чином, актуальність застосування OpenVPN та Wireguard полягає у їх здатності забезпечити надійний захист даних та високу продуктивність, що є критично важливим у контексті сучасних технологій розумних будинків.

1.2 Огляд існуючих рішень системи, що розробляється

У сучасному світі існує багато різних рішень для забезпечення захищеного віддаленого доступу до мереж, серед яких особливо виділяються комерційні VPN-роутери, програмні VPN-рішення на домашньому сервері або NAS, Хмарні VPN-сервери та OpenWRT на домашньому роутері. Ці технології забезпечують створення VPN, які захищають дані під час їх передачі через Інтернет.

Комерційні VPN-роутери є спеціалізованими пристроями, що мають вбудовану підтримку VPN. Вони забезпечують простий інтерфейс налаштувань і часто поставляються з технічною підтримкою від виробника.

VPN-з'єднання складається з VPN-сервера і VPN-клієнта (рис. 1.1). Якщо ви налаштовуєте маршрутизатор як VPN-сервер, вам потрібно встановити на кожному пристрої програмне забезпечення VPN-клієнта, щоб встановити VPN-з'єднання між маршрутизатором і вашими пристроями. У цьому випадку дані, що передаються між маршрутизатором і пристроями,

шифруються в приватному тунелі, забезпечуючи безпеку ваших даних. Якщо ви налаштуєте маршрутизатор як VPN-клієнт, то після підключення його до VPN-сервера всі підключені до нього пристрої зможуть користуватися послугою VPN, і вам не потрібно буде встановлювати програмне забезпечення VPN на кожен пристрій.

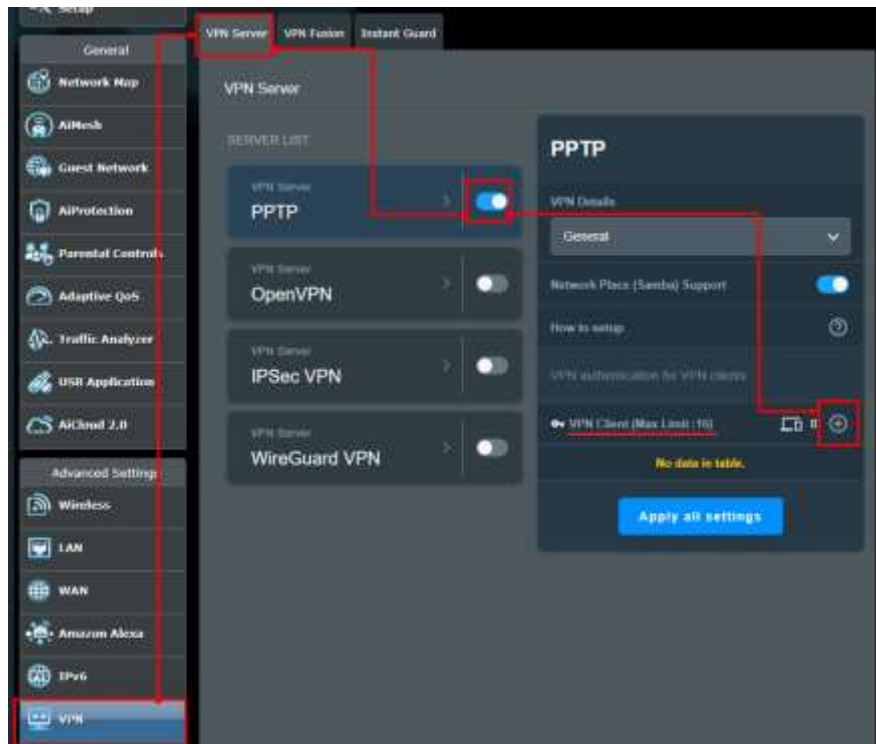


Рисунок 1.1 – Налаштування VPN серверу на роутері ASUS

Плюси комерційних VPN-роутерів:

- більшість комерційних VPN-роутерів мають простий інтерфейс налаштувань, що дозволяє швидко і легко налаштувати VPN;
- комерційні рішення часто мають потужніше апаратне забезпечення, що дозволяє підтримувати більше одночасних з'єднань і забезпечувати кращу продуктивність;
- виробники таких роутерів надають технічну підтримку, що може бути корисним у разі виникнення проблем;

Мінуси комерційних VPN-роутерів:

– комерційні VPN-роутери можуть бути дорогими у порівнянні з рішенням на базі RPI;

– деякі моделі можуть мати обмежені можливості налаштування у порівнянні з рішенням на базі RPI, яке можна налаштувати під специфічні потреби користувача;

Хмарні VPN-рішення надають можливість створення VPN через хмарного провайдера (рис. 1.2). Це дозволяє отримати доступ до VPN з будь-якого місця без необхідності мати статичну IP-адресу. Використовуючи хмарні обчислювальні служби без належних заходів хмарної безпеки, з'являється можливість для викрадення даних, знищення конфіденційних файлів і несанкціонованого віддаленого входу у вашу систему.

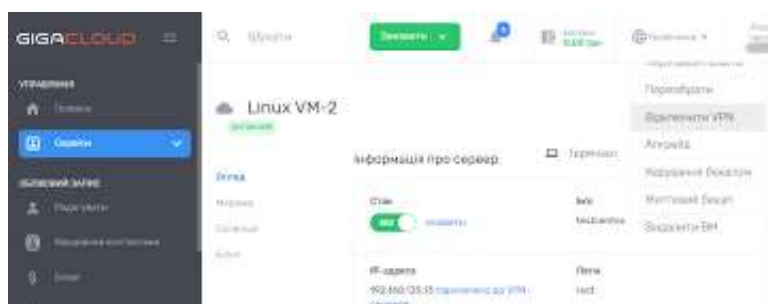


Рисунок 1.2 – Налаштування хмарного VPN сервісу

Плюси хмарних VPN-рішень:

– можливість доступу до VPN з будь-якого місця без необхідності мати статичну IP-адресу;

– хмарні провайдери зазвичай забезпечують високу доступність і надійність сервісів;

Мінуси хмарних VPN-рішень:

– постійні витрати на хмарні сервіси можуть накопичуватися з часом;

– залежність від третьої сторони для забезпечення доступу і безпеки даних;

OpenWRT – це прошивка для домашніх роутерів, яка дозволяє налаштувати VPN-сервер (рис. 1.3). Вона надає великі можливості

налаштування і може бути встановлена на багатьох моделях роутерів. OpenWRT надає повністю доступну для запису файловою системою з керуванням пакетами. Це звільняє вас від вибору та конфігурації додатків, що надаються виробником, і дозволяє вам налаштувати пристрій за допомогою пакетів під будь-яку програму. Для розробників OpenWRT є основою для створення додатків без необхідності створювати повну прошивку навколо них; для користувачів це означає можливість повної кастомізації, щоб використовувати пристрій у способах, які ніколи не передбачалися.

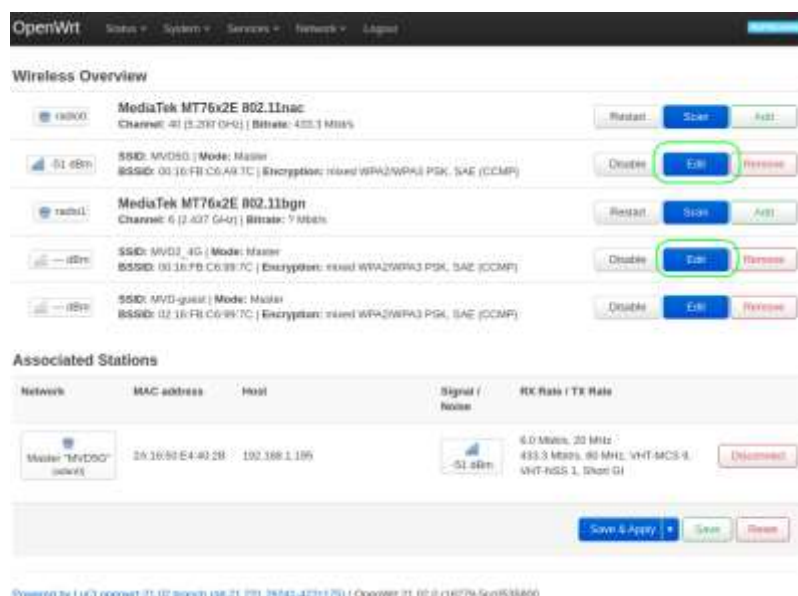


Рисунок 1.3 – Налаштування прошивки OpenWRT

Плюси використання OpenWRT:

- проєкт OpenWRT дозволяє налаштувати VPN-сервер на багатьох моделях домашніх роутерів з великими можливостями налаштування;
- використання існуючого роутера з прошивкою OpenWRT може бути дешевим рішенням;

Мінуси використання OpenWRT:

- налаштування і управління OpenWRT може бути складним для новачків;

– продуктивність VPN-сервера може бути обмежена апаратними можливостями домашнього роутера;

OpenVPN є зрілою і добре перевіреною технологією, яка широко використовується завдяки своїй гнучкості та безпеці. OpenVPN підтримує різні методи шифрування і автентифікації, що робить його універсальним рішенням для багатьох типів мереж. Вона дозволяє налаштувати тунельний режим, що захищає всі дані, що проходять через мережу, забезпечуючи високий рівень безпеки і конфіденційності.

Wireguard – це новіша технологія, яка здобула популярність завдяки своїй простоті та ефективності. Wireguard використовує сучасні криптографічні протоколи, що забезпечують високу продуктивність та менші затримки порівняно з іншими VPN-рішеннями. Її кодова база є компактнішою, що знижує ймовірність вразливостей і полегшує обслуговування[6].

Вебзастосунки для керування системами розумного будинку зазвичай забезпечують користувачам зручний інтерфейс для управління різними пристроями, такими як освітлення, системи опалення, охорони, та інші IoT пристрої. Ці застосунки можуть бути написані на різних мовах програмування і використовувати різні фреймворки та платформи[7].

У даному проєкті вебзастосунок для керування розумним будинком написаний на мові програмування Python. Python є популярною мовою завдяки своїй простоті і великій кількості бібліотек, що дозволяє швидко і ефективно розробляти різні додатки. Для створення вебзастосунку використовуються система Prometheus та сервіс Grafana[25].

Prometheus – це потужна система моніторингу і оповіщення з відкритим вихідним кодом, яка розроблена для збору і зберігання метрик в часовому ряду. Вона була створена компанією SoundCloud і з тих пір стала одним з найпопулярніших інструментів у своїй категорії. Основні особливості Prometheus включають збір, зберігання даних. Prometheus використовує потужну мову запитів PromQL для витягування та обробки даних. Prometheus має вбудовану систему оповіщень, яка дозволяє налаштовувати правила

оповіщень і відправляти їх через різні канали (email, Slack, PagerDuty тощо)[26].

Grafana – це популярна платформа для аналітики та візуалізації даних з відкритим вихідним кодом. Вона дозволяє створювати інтерактивні і привабливі інформаційні панелі, які можуть використовувати дані з різних джерел, включаючи Prometheus[27].

Prometheus і Grafana тісно інтегровані між собою, що дозволяє створювати ефективні системи моніторингу та візуалізації. Prometheus збирає метрики з різних кінцевих точок, які можуть бути сенсорами або іншими пристроями в системі розумного дому. Ці метрики зберігаються в базі даних Prometheus у вигляді часових рядів. Grafana підключається до Prometheus як до джерела даних. Для цього в Grafana налаштовується джерело даних з типом Prometheus і вказується URL-адреса сервера Prometheus. Комбінація Prometheus і Grafana забезпечує потужне рішення для моніторингу, аналізу та візуалізації даних у системах розумного дому та інших додатках[3].

Ідея проєкту полягає в інтеграції двох потужних технологій – VPN та вебзастосунків для розумних будинків, з використанням RPI як апаратної платформи. Унікальність цього проєкту полягає у декількох аспектах:

– **комплексний підхід до безпеки:** Використання як OpenVPN, так і Wireguard для перевірки здібностей цих технологій дозволяє знайти оптимальне рішення для забезпечення захищеного віддаленого доступу до систем домашньої автоматизації. Це дозволяє порівняти переваги та недоліки кожної з цих технологій у реальних умовах;

– **ефективність та продуктивність:** Завдяки використанню RPI, проєкт демонструє можливість створення недорогого і енергоефективного рішення для домашньої автоматизації. Це дозволяє забезпечити доступність таких систем для широкого кола користувачів;

– **гнучкість та масштабованість:** Вебзастосунок, написаний на Python, легко модифікується і розширюється, що дозволяє додавати нові функції та

інтегрувати додаткові пристрої у систему. Це робить рішення гнучким і здатним до адаптації під конкретні потреби користувачів;

– **простота налаштування та використання:** Всі компоненти системи, включаючи VPN сервер та вебзастосунок, налаштовані для максимальної простоти у використанні. Це знижує поріг входу для користувачів, які можуть не мати глибоких технічних знань;

Таким чином, проєкт поєднує у собі передові технології безпеки і зручність користування, що робить його унікальним та актуальним рішенням у сфері домашньої автоматизації[15].

1.3 Формування вимог до апаратно-програмного забезпечення

Основним апаратним компонентом є мікрокомп'ютер RPI. Використовується модель з достатньою потужністю та ресурсами для обробки мережевого трафіку і запуску необхідного програмного забезпечення, такі як RPI 3. [1]

Таблиця 1.1 – Характеристики Raspberry Pi 3

Процесор	Quad-core 64-bit ARM Cortex-A53 @ 1.2GHz
Графічний процесор	Broadcom VideoCore IV
Оперативна пам'ять	1GB LPDDR2 (900 MHz)
Порти USB	4 x USB 2.0
Порти Ethernet	10/100 Ethernet
Порти HDMI	Full-size HDMI
Живлення	5V/2.5A DC через micro-USB порт
Розміри	85.6 x 56.5 x 17 мм
Операційна система	Raspbian, а також підтримка інших ОС

MicroSD карта використовується для встановлення операційної системи та зберігання конфігураційних файлів. Рекомендована ємність - від 32 ГБ і вище.

Стабільне живлення з мінімальною потужністю 5В/3А для забезпечення стабільної роботи RPI та підключених пристроїв.

Вибір операційної системи впливає на стабільність і продуктивність системи, тому рекомендовано використовувати оптимізовану для RPI версію Linux – RPI OS. Для реалізації VPN сервера будуть використовуватися OpenVPN і Wireguard, які забезпечують високий рівень безпеки та гнучкості налаштувань. Вебзастосунок для управління розумним будинком буде написаний на мові програмування Python з використанням Prometheus та Grafana, що забезпечують структурованість і зручність у розробці серверної частини застосунку.

Функціональні вимоги визначають, які завдання система повинна виконувати, та включають опис функцій, необхідних для досягнення цілей проєкту. Вони охоплюють аспекти безпеки, зручності використання та продуктивності. Зокрема, система повинна забезпечувати шифрування даних, автентифікацію користувачів, інтуїтивно зрозумілий інтерфейс вебзастосунку для управління пристроями розумного будинку, а також стабільну та високу швидкість передачі даних через VPN канал[8].

Технічна підтримка та документація є важливими компонентами для забезпечення ефективного використання та обслуговування системи. Цей розділ включає надання детальної документації для користувачів та адміністраторів, яка пояснює процес налаштування, використання та вирішення можливих проблем. Також передбачено використання популярних програмних рішень із активною спільнотою підтримки, що допомагає швидко знайти відповіді на запитання та отримати допомогу у разі виникнення труднощів.

Вимоги до мережевої інфраструктури включають стабільне інтернет-з'єднання для забезпечення віддаленого доступу до системи через VPN, а також наявність локальної мережі для підключення RPI та всіх пристроїв розумного будинку. Це забезпечує надійне з'єднання і взаємодію між компонентами системи, необхідну для безперебійного функціонування.

1.4 Масштабованість та оптимізація продуктивності апаратно-програмного забезпечення

Кількість користувачів, які зможуть одночасно використовувати цей сервер, залежить від продуктивності апаратного забезпечення RPI та ефективності налаштування VPN. RPI 3 з 1 ГБ оперативної пам'яті може підтримувати від 2 до 5 одночасних користувачів при звичайних умовах використання. Якщо передбачається велике навантаження, може знадобитися використання додаткових апаратних ресурсів або налаштування для оптимізації продуктивності[2].

Для підтримки більшої кількості користувачів або додаткових функцій необхідно врахувати такі аспекти масштабованості системи:

- для збільшення кількості користувачів можна використовувати моделі RPI з більшим об'ємом оперативної пам'яті та потужнішими процесорами. Це дозволить обробляти більше одночасних з'єднань та забезпечити стабільну роботу системи;

- встановлення декількох RPI для розподілу навантаження між ними. Це можна зробити шляхом налаштування кластеру, що дозволить балансувати навантаження та забезпечувати резервування у разі відмови одного з пристроїв;

- використання рішень для балансування навантаження, таких як HAProxy або Nginx, дозволяє рівномірно розподіляти трафік між декількома інстанціями VPN серверів;

- тонке налаштування параметрів OpenVPN та Wireguard для підвищення ефективності обробки трафіку. Це включає оптимізацію шифрування, розподіл ресурсів та керування з'єднаннями;

- використання комутаторів та маршрутизаторів з високою пропускнуою здатністю для підтримки збільшеного мережевого трафіку;

– налаштування віддаленого доступу для адміністрування системи та впровадження резервних мережевих каналів для забезпечення безперервної роботи;

Для підтримки актуальності та безпеки системи необхідно регулярно проводити оновлення програмного забезпечення. Періодична перевірка наявності оновлень для RPI OS і встановлення важливих оновлень безпеки та покращень функціональності. Налаштування автоматичного завантаження та встановлення оновлень за допомогою системних інструментів, таких як `sudo` або спеціалізовані скрипти[8].

Слідкування за новими версіями OpenVPN та Wireguard, які можуть містити важливі виправлення безпеки та покращення продуктивності. Регулярне оновлення до останніх стабільних версій. Перевірка та оновлення конфігураційних файлів у відповідності до нових версій програмного забезпечення для забезпечення сумісності та оптимальної роботи.

Ці заходи дозволять забезпечити стабільну, безпечну та продуктивну роботу системи, а також підготувати її до масштабування відповідно до зростаючих вимог користувачів та додаткових функціональних потреб.

1.5 Технологічне рішення

VPN-сервер на базі RPI вирішує проблему безпечного та приватного доступу до домашньої мережі з будь-якого місця у світі. Це особливо важливо для захисту конфіденційних даних, забезпечення анонімності в Інтернеті та безпечного доступу до домашніх пристроїв. Використання RPI для цієї мети є економічно вигідним рішенням, оскільки цей невеликий, доступний і енергоефективний комп'ютер може забезпечити надійний VPN-сервіс з використанням OpenVPN або інших VPN-технологій.

RPI використовується для розгортання VPN-сервера за допомогою PiVPN та технології OpenVPN. Це дозволяє забезпечити безпечний доступ до вебзастосунку HAD з будь-якої точки світу через зашифроване з'єднання. Встановлення OpenVPN на RPI включає наступні етапи:

- вибір технології VPN та налаштування відповідних параметрів;
- вибір протоколу UDP для забезпечення швидкого та ефективного обміну даними;
- вибір DNS-провайдера Google для забезпечення швидкого та надійного DNS-резолвінгу;
- налаштування входу користувача через локальну IP-адресу RPI, що дозволяє підключатися до VPN без статичної IP-адреси;

Користувачі розумних будинків часто потребують доступу до своїх домашніх мереж з віддалених місць. Наприклад для віддаленого моніторингу домашніх систем. Використання VPN-сервера дозволяє створити захищене з'єднання між віддаленим пристроєм та домашньою мережею, забезпечуючи безпечний доступ до внутрішніх ресурсів, таких як файли, сенсорів або камери відеоспостереження.

Використання відкритих Wi-Fi мереж може бути небезпечним через ризик перехоплення даних. VPN-сервер на базі RPI шифрує весь інтернет-трафік, що проходить через нього, тим самим захищаючи особисті дані користувачів від потенційних загроз. VPN може бути використаний для доступу до сервісів, які можуть бути обмежені або заблоковані у певних регіонах. Підключаючись до домашнього VPN-сервера, користувач може отримати доступ до своїх улюблених веб-сайтів та онлайн-сервісів, начебто він знаходиться вдома.

Для додаткової зручності та функціональності була розроблена веб-застосунок HAD. Цей застосунок дозволяє користувачам моніторити та керувати різними аспектами свого розумного дому з будь-якої точки світу через захищене VPN-з'єднання. Він надає зручний інтерфейс для відображення важливих параметрів навколишнього середовища та споживання енергії, допомагаючи користувачам ефективніше керувати своїм домом.

Для створення та налаштування панелей вебзастосунку HAD використовується JSON. Кожна панель описана в JSON-коді, що містить конфігурацію джерела даних Prometheus, налаштування відображення полів, позицію на сітці, параметри відображення та цільові запити до Prometheus.

Мова програмування Python використовується для збирання даних з сенсорів, встановлених у розумному будинку. Скрипти на Python збирають дані з сенсорів CO₂, температури, вологості, шуму та електроенергії, використовуючи різні бібліотеки для взаємодії з апаратними інтерфейсами сенсорів. Зібрані дані передаються до Prometheus для подальшого зберігання та обробки.

Prometheus використовується як система моніторингу та зберігання часових рядів даних. Він збирає метрики від Python-скриптів через HTTP-запити та зберігає їх у вигляді часових рядів. Вебзастосунок HAD використовує Prometheus як джерело даних для візуалізації зібраних метрик на панелях, описаних у JSON-коді.

Панель CO₂ Level (Kitchen) відображає рівень вуглекислого газу у кухні. Моніторинг рівня CO₂ є важливим для підтримки здорового повітря в приміщенні, оскільки високий рівень CO₂ може негативно впливати на самопочуття та продуктивність людей. Панель використовує кольорові індикатори для швидкого візуального оцінювання стану повітря: зелений колір означає безпечний рівень, а червоний – небезпечний.

Панель Power Usage показує споживання енергії різними пристроями у вашому домі, зокрема нагрівачами. Вона допомагає користувачам відслідковувати та оптимізувати енергоспоживання, що може призвести до зниження витрат на електроенергію та зменшення екологічного впливу. Панель може включати в себе середні значення, максимальні значення та інші важливі статистичні дані про використання енергії.

Панель Temperature All відображає температурні та вологості параметри у різних приміщеннях дому, таких як кухня, ванна кімната, спальня і вітальня. Регулярний моніторинг цих параметрів допомагає забезпечити комфортні

умови для проживання та запобігати розвитку плісняви та інших негативних явищ, пов'язаних з неправильним рівнем вологості.

Панель Noise Level дозволяє відслідковувати рівень шуму у різних частинах дому. Моніторинг рівня шуму є важливим для підтримання комфортних умов проживання, особливо вночі або в періоди, коли потрібна тиша. Панель відображає дані з різних сенсорів, встановлених у ванній кімнаті, спальні, кухні та вітальні, дозволяючи швидко визначити джерела шуму та вжити відповідних заходів.

Висновки до розділу 1

Розділ "Аналіз предметної сфери" містить кілька ключових аспектів, які допомагають зрозуміти контекст і важливість розробки системи, що досліджується. Огляд існуючих рішень показав, що на ринку присутні різноманітні системи налаштувань VPN-серверів, які відрізняються функціональністю, ціною та складністю налаштування. Багато з них використовують хмарні сервіси для зберігання та аналізу даних, що може викликати питання безпеки та конфіденційності. Аналізуючи ці рішення, було визначено, що оптимальною є система, яка поєднує в собі простоту використання, високу функціональність та безпеку.

Таким чином, у розділі було визначено, що створення VPN-серверу на базі RPI для роботи розробленого вебзастосунку керування панелі розумного дому є актуальною і важливою для сучасного суспільства. Проведений огляд існуючих рішень вказав на необхідність створення системи, яка б відповідала високим вимогам безпеки та зручності використання. Сформовані вимоги до апаратно-програмного забезпечення закладають основу для подальшого проектування та реалізації ефективної та надійної системи мережевої безпеки вебзастосунку керування панелі розумного дому.

2 ПРОЄКТУВАННЯ ЗАХИЩЕНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ З ВИКОРИСТАННЯМ VPN СЕРВЕРІВ

2.1 Налаштування VPN сервера

Для створення VPN-сервера часто використовуються спеціалізовані апаратні рішення або потужні сервери. Однак, із розвитком мініатюрних та доступних одноплатних комп'ютерів, таких як RPI, з'явилася можливість реалізувати VPN-сервер на їх основі. RPI є універсальною платформою, яка поєднує в собі компактні розміри, низьку вартість та достатню обчислювальну потужність для виконання різних завдань, включаючи роль VPN-сервера.

RPI 3, зокрема, є одним з найпопулярніших варіантів цієї платформи завдяки своїй продуктивності та широким можливостям підключення. Він оснащений процесором Broadcom BCM2837, має 1 ГБ оперативної пам'яті, чотири USB порти для підключення периферійних пристроїв, Ethernet порт для проводового підключення до мережі, а також вбудовані модулі Wi-Fi та Bluetooth.

Схема RPI 3 містить всі необхідні компоненти для створення ефективного та надійного VPN-сервера. У цій роботі буде детально розглянуто архітектуру RPI 3, описано його основні компоненти та можливості, а також наведено приклад використання цієї платформи для розгортання VPN-сервера.

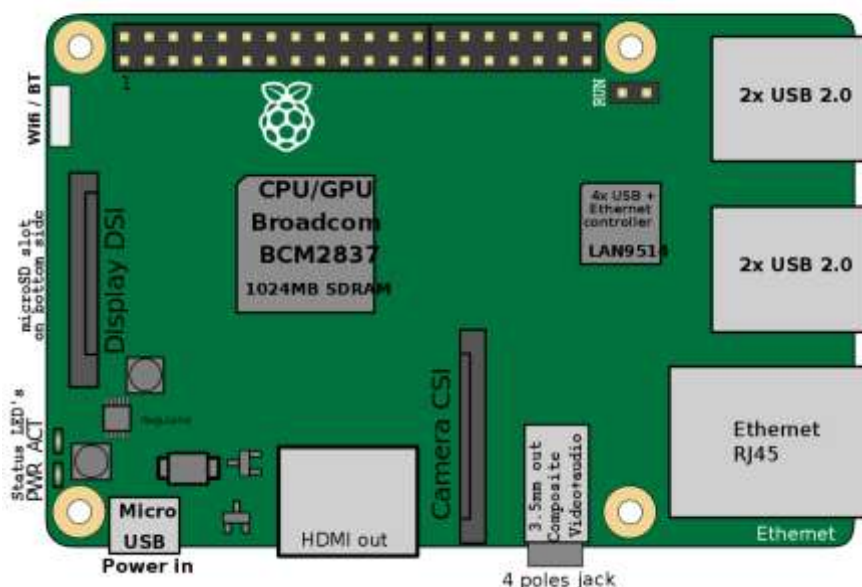


Рисунок 2.1 – Технічна схема Raspberry PI 3

Можна виділити такі основні компоненти та інтерфейси на рис. 2.1:

- центральний процесор Broadcom BCM2837 з тактовою частотою 1.2 ГГц і 1 ГБ оперативної пам'яті SDRAM, що забезпечують достатню обчислювальну потужність для різноманітних завдань;
- чотири USB 2.0 порти для підключення периферійних пристроїв, таких як клавіатура, миша або зовнішній накопичувач, та Ethernet порт для підключення до мережі Інтернет;
- порт HDMI для підключення монітора або телевізора, що дозволяє використовувати RPI як мультимедійний центр;
- комбінований 3.5 мм роз'єм для виводу аудіо та композитного відео;
- роз'єм Micro USB для підключення джерела живлення;
- інтерфейси DSI та CSI для підключення дисплеїв та камер відповідно;
- слот для microSD карт, який використовується для завантаження операційної системи та зберігання даних;
- вбудовані модулі Wi-Fi та Bluetooth для бездротового підключення;

Дана конфігурація робить RPI 3 ідеальним кандидатом для створення VPN-сервера, який забезпечить захищене та надійне з'єднання з Інтернетом.

Далі буде детально розглянуто процес налаштування та оптимізації VPN-сервера на базі RPI 3.

Налаштування VPN сервера на базі RPI є ефективним способом забезпечення безпечного доступу до мережі з віддалених місць. RPI, як доступний та компактний пристрій, є ідеальною платформою для реалізації VPN сервера. Встановлення та конфігурація VPN сервера за допомогою PiVPN дозволяє спростити процес налаштування і забезпечити надійну роботу мережі.

Процес починається з підготовки RPI, що включає встановлення останньої версії операційної системи RPI OS. Після цього необхідно виконати оновлення системи та інсталяцію всіх необхідних пакетів. Зокрема, використовується PiVPN - зручний інструмент для швидкого налаштування OpenVPN або WireGuard, які забезпечують високий рівень безпеки та продуктивності[9].

Під час інсталяції PiVPN користувачеві пропонується вибрати між OpenVPN та WireGuard. Обидві технології мають свої переваги: OpenVPN забезпечує широку сумісність та гнучкість налаштувань, тоді як WireGuard відомий своєю простотою та високою продуктивністю. Після вибору потрібної технології PiVPN автоматично завантажує та інсталує відповідне програмне забезпечення.

Після завершення інсталяції починається процес налаштування серверу. PiVPN пропонує користувачеві покрокові інструкції, що включають налаштування мережевих параметрів, генерацію ключів та сертифікатів, а також створення користувацьких профілів для підключення клієнтів. Користувачі можуть створювати та управляти VPN профілями за допомогою простих команд у терміналі[11].

Завершальний етап включає перевірку налаштувань та запуск VPN сервера. PiVPN автоматично налаштовує необхідні правила firewall та включає IP форвардинг, забезпечуючи коректне функціонування VPN. Після цього VPN сервер готовий до використання, і користувачі можуть підключатися до

нього за допомогою створених профілів, отримуючи безпечний доступ до локальної мережі з будь-якого місця.

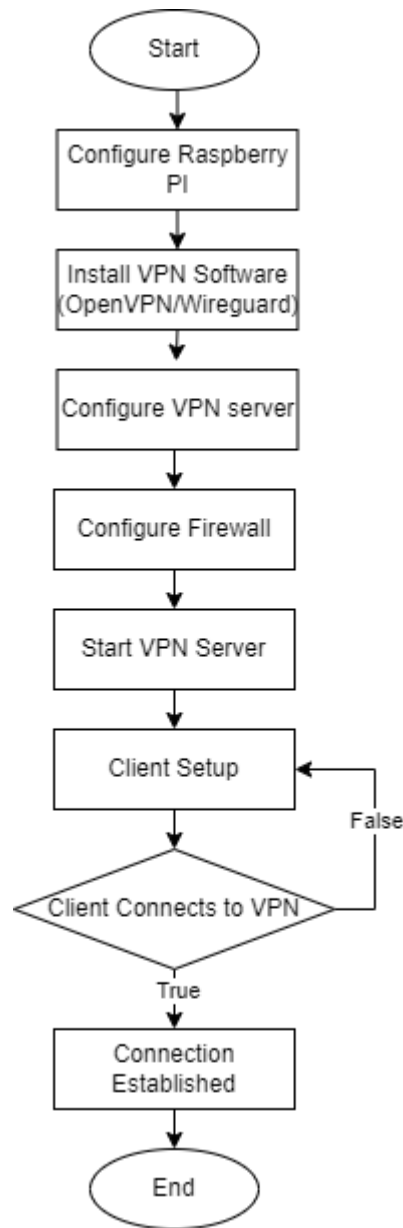


Рисунок 2.2 – Блок-схема налаштування VPN серверу

Процес налаштування VPN сервера на базі RPI за допомогою інструменту PiVPN демонструється на рис 2.2. Блок-схема покроково описує всі необхідні етапи, від підготовки RPI до кінцевого налаштування і запуску VPN сервера. Схема допоможе візуально зрозуміти основні кроки, включаючи встановлення операційної системи, оновлення системи, інсталяцію PiVPN, вибір технології VPN, налаштування мережевих параметрів, а також

тестування та запуск сервера (рис. 2.2). Процес налаштування забезпечить надійне та безпечне з'єднання для віддалених користувачів, які потребують доступу до локальної мережі[12].

У даній роботі розглянуто створення вебзастосунку комплексної системи моніторингу розумного будинку з використанням RPI 3 як VPN-сервера, який забезпечить захищене з'єднання, збір та передачу даних з сенсорів, їх збереження у системі Prometheus та візуалізацію в реальному часі за допомогою Grafana.

Процес розпочинається зі старту VPN-сервера на RPI, що дозволяє користувачам підключатися до мережі розумного будинку через захищений канал. Це забезпечує високий рівень безпеки під час передачі даних з сенсорів розумного будинку.

Дані з сенсорів розумного будинку збираються за допомогою Python-скрипту, що працюють на RPI. Цей скрипт відповідають за форматування та передачу даних до системи Prometheus, яка забезпечує їх збереження та подальший аналіз.

Prometheus приймає дані від Python-скрипту, зберігає їх у своїй базі даних та забезпечує їх актуальність шляхом періодичного зчитування даних з сенсорів. Це дозволяє створити надійну систему моніторингу, що забезпечує своєчасну обробку даних.

Grafana, підключена до Prometheus, отримує збережені дані та виводить їх на панелі дашборду. Використовуючи JSON код, ми налаштовуємо дашборд, визначаючи структуру панелей, джерела даних та типи візуалізацій. Це дозволяє користувачам бачити в реальному часі всі дані з сенсорів розумного будинку у вигляді графіків, діаграм та інших візуальних елементів.

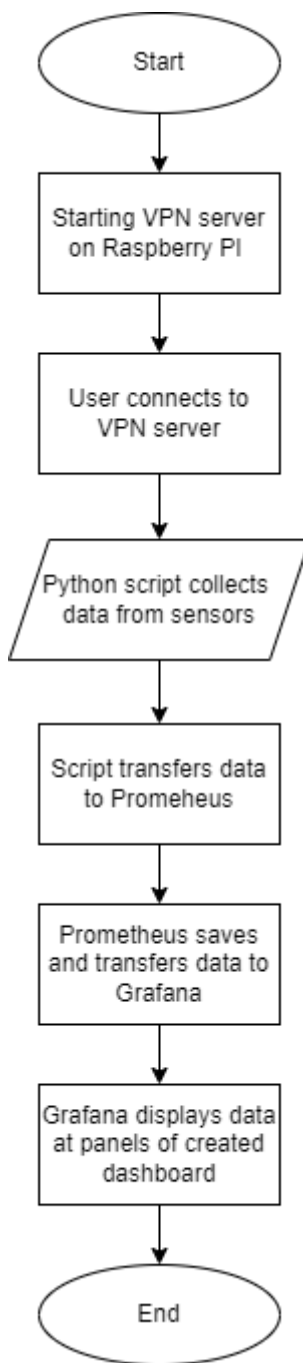


Рисунок 2.3 – Блок-схема роботи вебзастосунку Home Automation Dashboard

Процес від моменту підключення користувача до VPN-сервера, збору та передачі даних з сенсорів, до їх обробки та візуалізації в Grafana детально описаний на рис. 2.3. У подальших розділах дипломної роботи будуть розглянуті всі етапи реалізації цієї системи, зокрема налаштування VPN-сервера, розробку Python-скрипту, конфігурація Prometheus та створення дашбордів у Grafana.

2.2 Налаштування мережевого моста з OpenVPN

Ethernet-міст по суті передбачає об'єднання інтерфейсу Ethernet з одним або декількома віртуальними інтерфейсами TAP і з'єднання їх разом під егідою єдиного інтерфейсу моста. Мости Ethernet являють собою програмний аналог фізичного комутатора Ethernet. Ethernet-міст можна розглядати як різновид програмного комутатора, який можна використовувати для з'єднання декількох Ethernet-інтерфейсів на одному комп'ютері, використовуючи при цьому одну IP-підмережу[23].

З'єднавши фізичну мережеву карту Ethernet з інтерфейсом TAP, керованим OpenVPN, у двох різних місцях, можна логічно об'єднати обидві мережі Ethernet так, ніби вони є однією підмережею Ethernet.

При використанні мостової конфігурації ethernet першим кроком є створення ethernet-мосту - різновиду віртуального мережевого інтерфейсу, який є контейнером для інших ethernet-інтерфейсів, або реальних, як у фізичних мережевих картах, або віртуальних, як у TAP-інтерфейсах. Інтерфейс моста Ethernet має бути налаштований до фактичного запуску OpenVPN.

Не існує універсального методу створення мостового інтерфейсу Ethernet - кожна ОС має свій власний метод.

Після створення мостового інтерфейсу і додавання до нього відповідних інтерфейсів Ethernet можна запускати OpenVPN[15].

Мостовий інтерфейс – це різновид віртуального мережевого інтерфейсу, який утворюється шляхом об'єднання одного або декількох інтерфейсів Ethernet, кожен з яких може бути фізичною мережевою картою або віртуальним інтерфейсом TAP, що використовується для тунелювання VPN.

Коли ви налаштовуєте ethernet-міст, ви повинні вручну задати IP-адресу і підмережу інтерфейсу моста, а не використовувати директиву `ifconfig` в конфігурації OpenVPN. Це пов'язано з тим, що на відміну від інтерфейсу

TUN/TAP, OpenVPN не може програмно встановити IP-адресу і маску підмережі інтерфейсу моста[20].

У конфігурації OpenVPN у директиві `devdirective` слід вказати компонент інтерфейсу TAP мостового інтерфейсу, а не ім'я самого мостового інтерфейсу.

У Windows використовуйте директиву `dev-node`, щоб назвати адаптер TAP-Win32, який було додано до мосту.

У Linux/BSD/Unix для директиви `dev tap` використовуйте явний номер пристрою TUN/TAP, який ви додали до мосту, наприклад, `dev tap0`.

Якщо ви використовуєте OpenVPN в режимі точка-точка, опустите директиву `ifconfig`, а якщо ви використовуєте режим клієнт-сервер, використовуйте на сервері директиву `server-bridge`.

При мостуванні ви повинні вручну встановити налаштування TCP/IP на інтерфейсі моста. Наприклад, у Linux це можна зробити за допомогою команди `ifconfig`.

Переконайтеся, що інтерфейси TAP з'єднано лише з приватними інтерфейсами Ethernet, які захищено брандмауером. Ніколи не з'єднуйте інтерфейс TAP з тим самим інтерфейсом Ethernet, який ви використовуєте для підключення до Інтернету, оскільки це може створити потенційну діру в безпеці.

Адреси, що використовуються для локального і віддаленого доступу, не повинні входити до складу мостової підмережі - інакше ви отримаєте петлю маршрутизації.

Важливим моментом, який слід розуміти при мостуванні Ethernet, є те, що кожен мережевий інтерфейс, який додається до мосту, втрачає свою індивідуальну ідентичність з точки зору специфічних налаштувань, таких як IP-адреса і маска підмережі. Релевантними будуть лише налаштування TCP/IP самого інтерфейсу моста.

Поширеною помилкою, якої припускаються при ручному налаштуванні моста Ethernet, є те, що вони додають до моста первинний адаптер Ethernet до

того, як встановили IP-адресу і маску мережі інтерфейсу моста. В результаті первинний інтерфейс Ethernet «втрачає» свої налаштування, але еквівалентні налаштування інтерфейсу моста ще не були визначені, тому кінцевим ефектом є втрата зв'язку на інтерфейсі Ethernet[19].

У більшості випадків можна налаштувати конфігурацію моста, коли сам ethernet-міст налаштовано лише на стороні сервера, а не на стороні клієнта. Якщо це зробити, клієнтські машини стануть мультидоменими при підключенні до сервера, тобто вони все ще матимуть свій звичайний інтерфейс Ethernet, але при підключенні до сервера OpenVPN вони матимуть новий інтерфейс TAP, який з'єднаний з інтерфейсом Ethernet сервера.

```
#!/bin/bash

#####
# Set up Ethernet bridge on Linux
# Requires: bridge-utils
#####

# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged,
# for example tap="tap0 tap1 tap2".
tap="tap0"

# Define physical ethernet interface to be bridged
# with TAP interface(s) above.
eth="eth0"
eth_ip="192.168.8.4"
eth_netmask="255.255.255.0"
eth_broadcast="192.168.8.255"
```

Рисунок 2.4(a) – Скрипт bridge-start

```
for t in $tap; do
    openvpn --mktun --dev $t
done

brctl addbr $br
brctl addif $br $eth

for t in $tap; do
    brctl addif $br $t
done

for t in $tap; do
    ifconfig $t 0.0.0.0 promisc up
done

ifconfig $eth 0.0.0.0 promisc up

ifconfig $br $eth_ip netmask $eth_netmask broadcast $eth_broadcast
```

Рисунок 2.4(б) – Скрипт bridge-start

Скрипт bridge-start (рис. 2.4) обробляє налаштування мосту у операційній системі Linux[22].

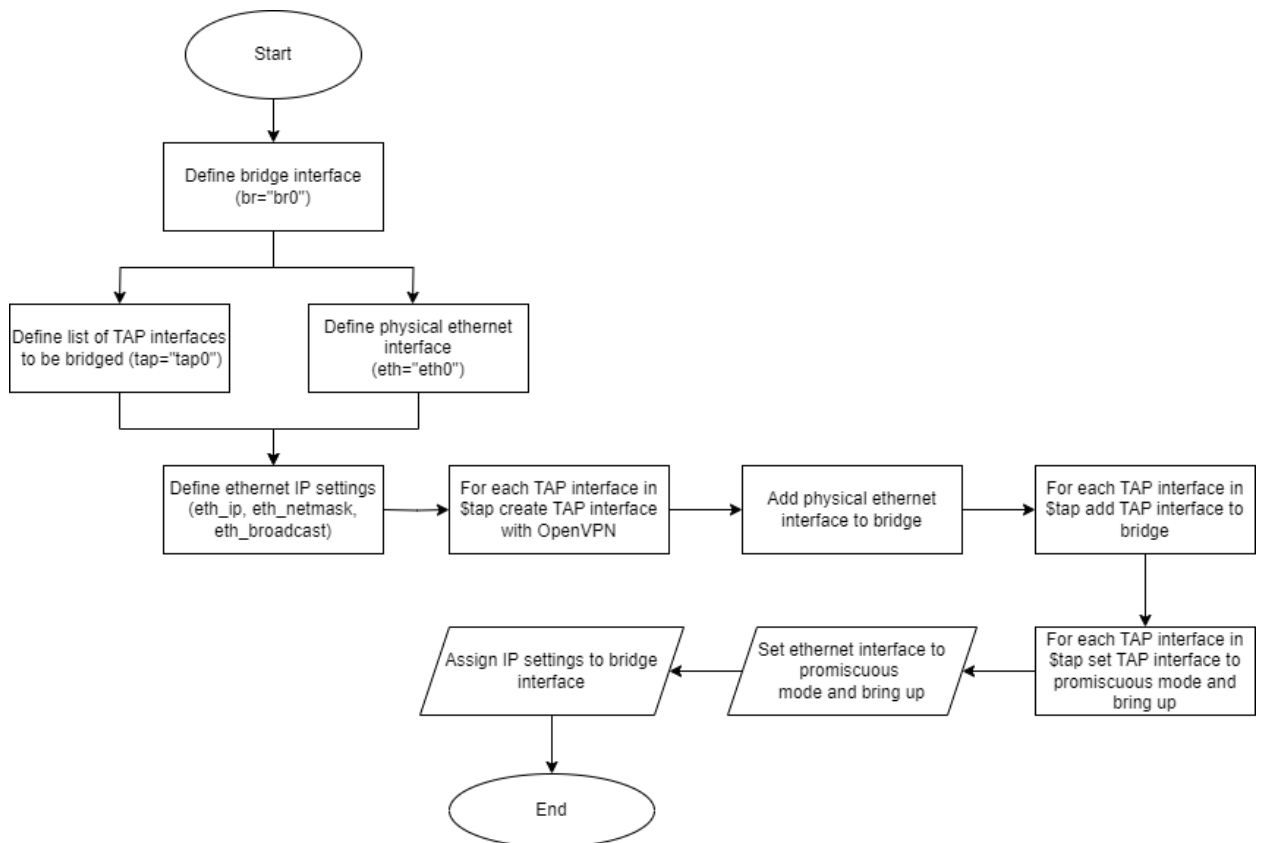


Рисунок 2.5 – Блок-схема налаштування мосту bridge-start

Процес налаштування Ethernet моста з використанням bash скрипту bridge-start, включаючи створення TAP інтерфейсів, додавання їх до мостового інтерфейсу, та налаштування IP конфігурації ілюструється на рис 2.5 [22].

Скрипт bridge-stop обробляє видалення мосту у Linux.

```
#!/bin/bash

#####
# Tear Down Ethernet bridge on Linux
#####

# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged together
tap="tap0"

ifconfig $br down
brctl delbr $br

for t in $tap; do
    ip netns exec --netns --dev $t
done
```

Рисунок 2.6 – Скрипт bridge-stop

Блок-схема алгоритму bridge-stop описує дії виконання bash скрипту(рис. 2.6).

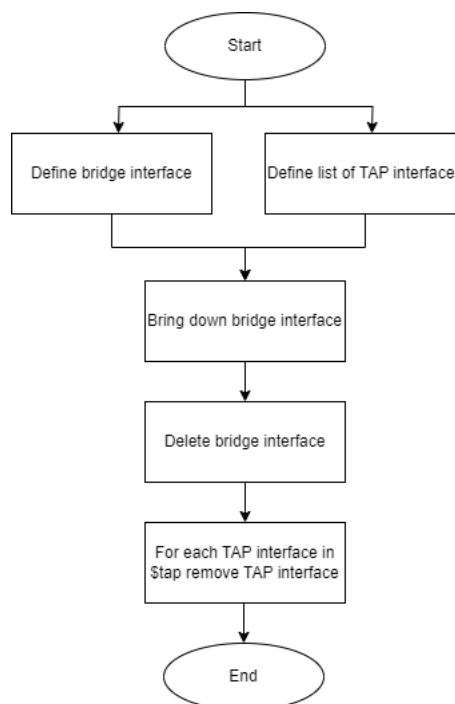


Рисунок 2.7 – Блок-схема налаштування мосту bridge-stop

Процес видалення Ethernet моста з використанням bash скрипту, включаючи вимкнення і видалення мостового інтерфейсу, а також видалення TAP інтерфейсів ілюструється на рис. 2.7.

2.3 Криптографічні алгоритми шифрування

Алгоритм AES (рис. 2.8) є одним з найбільш популярних і широко використовуваних симетричних блокових шифрів у сучасній криптографії. Він використовується в різних додатках для забезпечення конфіденційності та цілісності даних, включаючи технологію OpenVPN.

AES був розроблений як заміна для застарілого стандарту DES і затверджений як стандарт шифрування Федеральним інститутом стандартів і технологій США (NIST) у 2001 році. Він характеризується такими основними параметрами:

- блочний шифр AES обробляє дані блоками по 128 біт;
- ключі різної довжини підтримуються ключі довжиною 128, 192 і 256 біт, що визначають рівень безпеки;
- симетричний шифр, однаковий ключ використовується для шифрування та дешифрування даних;

Алгоритм AES складається з кількох основних компонентів, кожен з яких грає важливу роль у забезпеченні безпеки шифрування. Процес генерування серії раундових ключів з початкового шифрувального ключа називається – ключовий розклад. Ці раундові ключі використовуються в кожному етапі шифрування та дешифрування.

AES виконує ряд раундів (10, 12 або 14 залежно від довжини ключа), кожен з яких складається з чотирьох основних операцій:

- subBytes: Байти даних замінюються на основі фіксованої таблиці замін, що забезпечує нелінійність шифру;
- shiftRows: Байти в кожному рядку блоку даних зміщуються на різні позиції, що створює дифузю;

– mixColumns: Стовпці даних перемішуються, використовуючи лінійні перетворення, щоб додатково посилити дифузію;

– addRoundKey: Кожен байт блоку даних XOR-ується з відповідним байтом раундового ключа;

Фінальний раунд виконує ті ж операції, що й звичайні раунди, але без MixColumns.

У технології OpenVPN AES використовується для забезпечення конфіденційності переданих даних. Під час встановлення з'єднання OpenVPN генерує сесійний ключ, який використовується для шифрування даних за допомогою AES. Цей ключ генерується за допомогою алгоритмів обміну ключами, таких як Diffie-Hellman або ECDH, що забезпечує захищений обмін ключами між клієнтом і сервером[20].

OpenVPN підтримує різні режими роботи AES (рис. 2.5), включаючи CBC і GCM. Режим CBC забезпечує високу безпеку, але вимагає додаткової обробки для забезпечення цілісності даних, тоді як режим GCM є аутентифікованим шифром, що забезпечує одночасно конфіденційність і цілісність даних.

AES вважається дуже безпечним завдяки своїй стійкій конструкції та великому розміру ключів. Відомі атаки, такі як атаки на базі часу і атаки боковими каналами, можуть бути ефективними лише в умовах фізичного доступу до пристрою, але з використанням стандартних криптографічних практик ці ризики можуть бути мінімізовані.

Таким чином, AES забезпечує надійне шифрування даних в OpenVPN, що є ключовим для захисту інформації, що передається через VPN-з'єднання. Завдяки своїй гнучкості та високому рівню безпеки, AES є невід'ємною частиною сучасних криптографічних рішень.

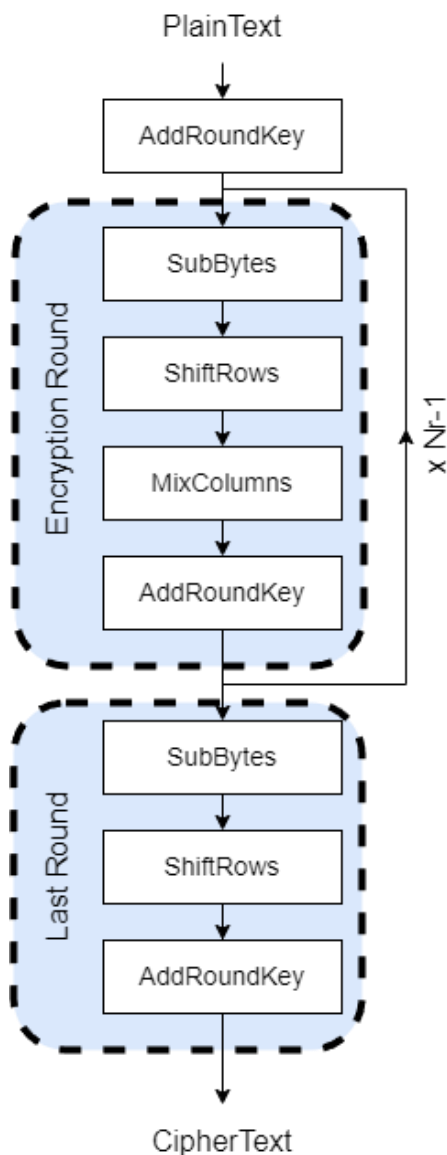


Рисунок 2.8 – Схема шифрування криптографічного алгоритму AES

Алгоритм ChaCha20 є одним із сучасних потокових шифрів, відомих своєю високою продуктивністю та безпекою. Він був розроблений Деніелом Дж. Бернштейном як покращена версія алгоритму Salsa20 і широко використовується в різних криптографічних додатках, включаючи VPN технологію WireGuard.

ChaCha20 (рис. 2.9) є потоковим шифром, що означає, що він шифрує дані побітно або побайтово, а не блоками, як це роблять блокові шифри. Основні характеристики ChaCha20 включають:

- потоковий шифр обробляє дані потоком, що забезпечує високу швидкість шифрування і зручність для роботи з даними довільної довжини;
- ключова довжина 256-бітовий ключ, що забезпечує високий рівень безпеки;
- ініціалізаційний вектор (IV) – 96-бітовий або 64-бітовий, який разом із ключем використовується для забезпечення унікальності шифрування кожного блоку даних.
- безпека: розроблений для стійкості до відомих криптографічних атак, таких як диференціальний криптоаналіз;

ChaCha20 побудований на основі простої, але ефективної конструкції.

Основні елементи включають:

- 256-бітовий ключ складається з восьми 32-бітових слів;
- Ініціалізаційний вектор (nonce) – це 96-бітовий nonce складається з трьох 32-бітових слів;
- лічильник, 32-бітове слово, яке збільшується для кожного блоку даних, що шифрується;

Алгоритм використовує ці параметри для генерування 512-бітового потоку псевдовипадкових чисел, який потім XOR-ується з відкритим текстом для отримання шифротексту.

ChaCha20 складається з 20 раундів операцій над 16-словним станом, який ініціалізується ключем, nonce і лічильником. Основні операції включають:

- quarter Round: базова операція, що виконується кілька разів у кожному раунді, включає додавання, XOR і циклічні зрушення (rotate);
- column Round і Diagonal Round: ці операції забезпечують перемішування даних у стовпцях і діагоналях матриці стану, що сприяє дифузії;

У WireGuard алгоритм ChaCha20 використовується для забезпечення конфіденційності переданих даних. WireGuard обрав ChaCha20 через його

високу продуктивність і безпеку, особливо на мобільних пристроях та інших платформах з обмеженими ресурсами. ChaCha20 працює швидше, ніж AES, на багатьох сучасних процесорах, оскільки не вимагає апаратної підтримки для досягнення високої продуктивності[10].

WireGuard використовує ChaCha20 у поєднанні з Poly1305 для забезпечення аутентифікації даних. Ця комбінація забезпечує як конфіденційність, так і цілісність даних, переданих через VPN. Кожен пакет даних шифрується за допомогою ChaCha20 і перевіряється на автентичність за допомогою Poly1305, що гарантує, що дані не були змінені під час передачі.

ChaCha20 вважається дуже безпечним алгоритмом завдяки своїй стійкій конструкції. Він витримав численні криптографічні аналізи і не має відомих вразливостей, які б могли бути використані для компрометації зашифрованих даних. Його структура і операції були розроблені з урахуванням стійкості до широкого спектру атак, включаючи атак на базі часу і атак боковими каналами.

Завдяки своїй швидкості, ефективності та безпеці, ChaCha20 є ідеальним вибором для VPN технологій, таких як WireGuard, які потребують швидкого та надійного шифрування даних для забезпечення захищених комунікацій[14].

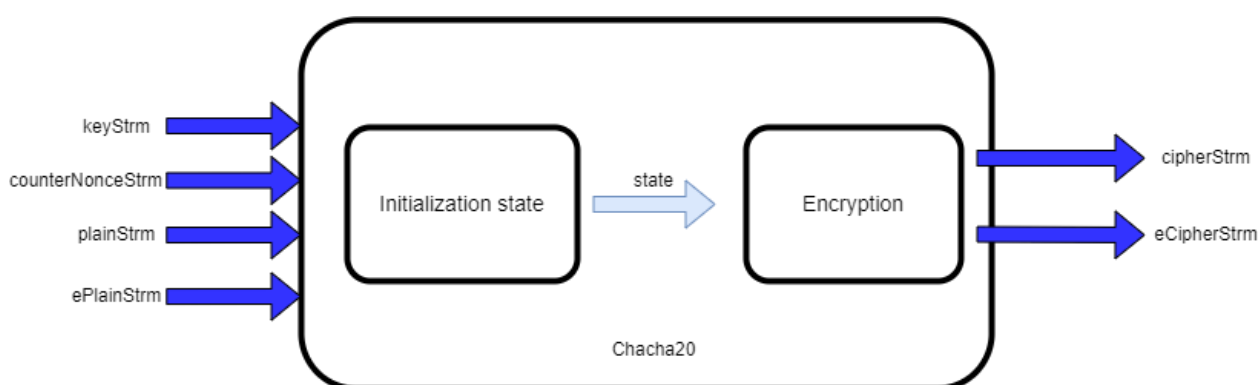


Рисунок 2.9 – Схема шифрування криптографічного алгоритму Chacha20

Алгоритми AES і ChaCha20 є двома широко використовуваними методами шифрування, кожен з яких має свої переваги та недоліки в залежності від конкретного випадку використання.

AES є симетричним блоковим шифром, що шифрує дані блоками по 128 біт з використанням ключів довжиною 128, 192 або 256 біт. Його головні переваги включають високу швидкість шифрування на пристроях з апаратною підтримкою AES та високу безпеку, підтверджену численними аналізами. Проте, його ефективність може знижуватись на пристроях без такої апаратної підтримки, особливо на мобільних пристроях або вбудованих системах.

ChaCha20, з іншого боку, є потоковим шифром, який шифрує дані побайтово з використанням 256-бітового ключа. ChaCha20 відомий своєю високою швидкістю та ефективністю, особливо на платформах без апаратної підтримки AES. Він також забезпечує високий рівень безпеки і є стійким до широкого спектру криптографічних атак. Завдяки своїй продуктивності на різних пристроях, ChaCha20 є популярним вибором для мобільних додатків і VPN технологій, таких як WireGuard.

Загалом, вибір між AES і ChaCha20 залежить від конкретних вимог до продуктивності та апаратних можливостей. AES є оптимальним вибором для середовищ з апаратною підтримкою, тоді як ChaCha20 є більш ефективним на платформах з обмеженими ресурсами, забезпечуючи при цьому надійний рівень безпеки.

2.4 Підключення користувача до VPN серверу

Після налаштування VPN сервера на RPI користувачі можуть підключитися до нього з будь-якого пристрою, що підтримує VPN. Для цього потрібно виконати кілька кроків [5]:

- встановлення VPN клієнта: на пристрої користувача потрібно встановити VPN клієнтське програмне забезпечення. Для OpenVPN можна використовувати офіційний клієнт OpenVPN, доступний для більшості операційних систем. Для WireGuard можна встановити клієнт WireGuard;

- імпорт конфігураційного файлу: у VPN клієнті потрібно імпортувати отриманий конфігураційний файл. Це можна зробити за допомогою

графічного інтерфейсу клієнта або командного рядка, залежно від обраного програмного забезпечення;

– підключення до VPN: після імпорту конфігураційного файлу користувач може ініціювати підключення до VPN серверу. Для цього потрібно відкрити VPN клієнт та натиснути кнопку "Підключитися". Клієнт встановить з'єднання з VPN сервером на RPI, використовуючи параметри з конфігураційного файлу;

Після успішного підключення користувач може перевірити, чи з'єднання встановлено правильно. Після встановлення з'єднання користувач може безпечно користуватися Інтернетом та доступом до локальної мережі[15].

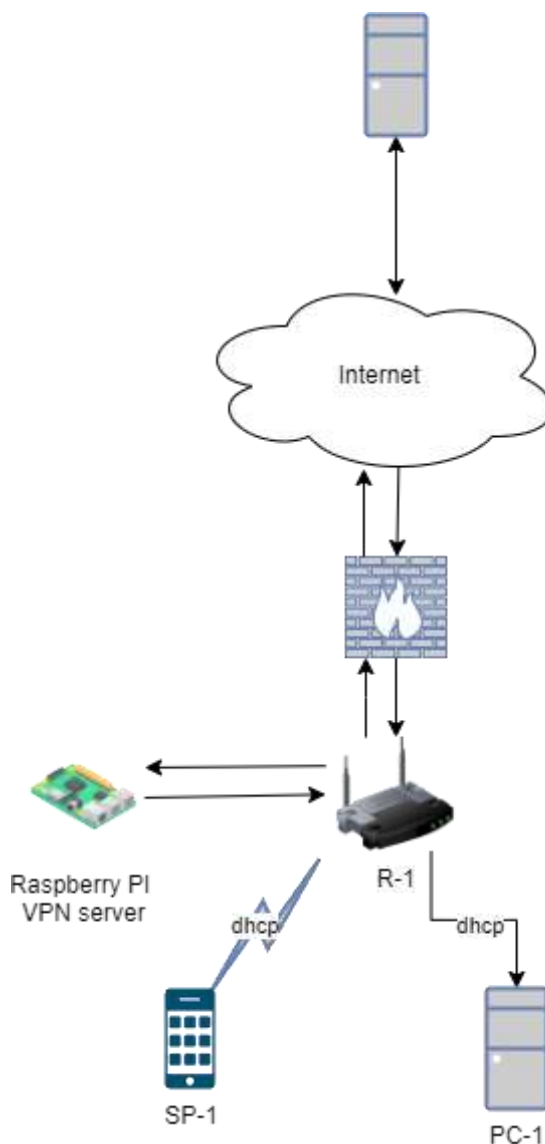


Рисунок 2.10 – Схема підключення пристроїв до VPN серверу

Завдяки налаштованому VPN серверу на базі RPI (рис. 2.10) користувачі можуть мати безпечний та захищений доступ до Інтернету та внутрішніх мережевих ресурсів з будь-якої точки світу.

Розглянемо підключення користувача до VPN-серверу за допомогою UML-діаграми (рис. 2.11). Інтернет приймає та передає зашифрований трафік між користувачем і VPN сервером, а також пересилає розшифрований трафік від VPN серверу до цільових ресурсів і назад. Цільові ресурси отримують запити через захищене з'єднання та відповідають на них, забезпечуючи користувачеві доступ до необхідної інформації та сервісів.

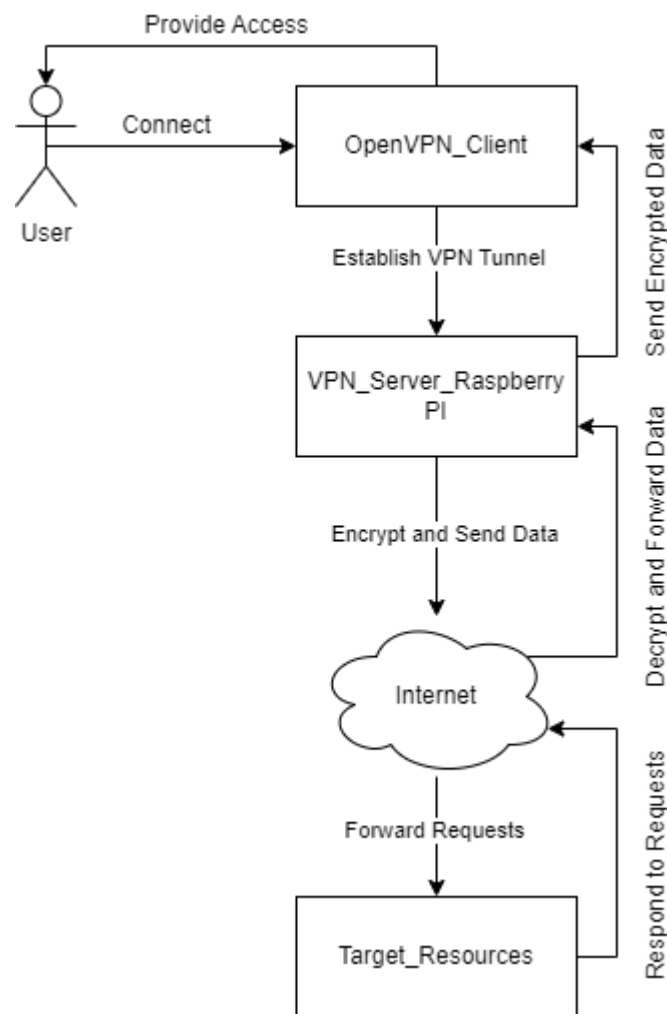


Рисунок 2.11 – UML-діаграма підключення користувача

Ця діаграма демонструє всі ключові етапи та компоненти процесу підключення, включаючи автентифікацію, шифрування трафіку, та передачу

даних через Інтернет, що забезпечує безпечний та надійний доступ до ресурсів.

Висновки до розділу 2

У розділі було детально розглянуто налаштування VPN-сервера, створення мережевого моста з OpenVPN, криптографічні алгоритми шифрування AES та Chacha20, а також підключення користувачів до VPN-сервера[17].

Налаштування VPN-сервера є критично важливим для забезпечення безпечного доступу до домашньої мережі. Використання VPN дозволяє захистити дані від несанкціонованого доступу та забезпечити конфіденційність комунікацій. У цьому контексті було налаштовано OpenVPN, який забезпечує надійне шифрування та автентифікацію користувачів.

Мережевий міст з OpenVPN дозволяє об'єднати декілька мереж в одну, що забезпечує безшовний доступ до всіх ресурсів мережі з будь-якого місця. Це дає можливість користувачам дистанційно керувати пристроями розумного будинку та отримувати доступ до них, як якщо б вони перебували вдома.

У розділі також було розглянуто криптографічні алгоритми шифрування AES та Chacha20, які використовуються для захисту даних. AES є широко вживаним стандартом шифрування, що забезпечує високу безпеку даних завдяки використанню симетричного ключа. Chacha20 є альтернативним алгоритмом, що пропонує високу швидкість шифрування та низьку затримку, що є особливо важливим для мобільних пристроїв та середовищ з обмеженими ресурсами.

Підключення користувачів до VPN-сервера було здійснено з урахуванням вимог до безпеки та зручності. Було налаштовано процедуру автентифікації та авторизації користувачів, що дозволяє контролювати доступ до мережевих ресурсів та забезпечити безпеку комунікацій[18].

Таким чином, виконана робота демонструє важливість використання сучасних криптографічних методів та налаштувань мережевої інфраструктури для забезпечення безпечного та ефективного функціонування системи розумного будинку. Застосування VPN та криптографічних алгоритмів дозволяє створити захищене середовище для передачі даних, що є необхідним для надійного керування та моніторингу розумного будинку.

3 РОЗРОБЛЕННЯ АПАРАТНО-ПРОГРАМНОЇ ЧАСТИНИ ПАНЕЛЬ КЕРУВАННЯ РОЗУМНИХ БУДИНКІВ

3.1 Налаштування VPN серверу на Raspberry PI

Для налаштування VPN серверу на RPI було використано PiVPN, що є простим у використанні скриптом для встановлення і конфігурації OpenVPN. Технологія OpenVPN була обрана для вебзастосунку HAD через її надійність, безпеку та широке використання в різних сценаріях мережевої безпеки[27].

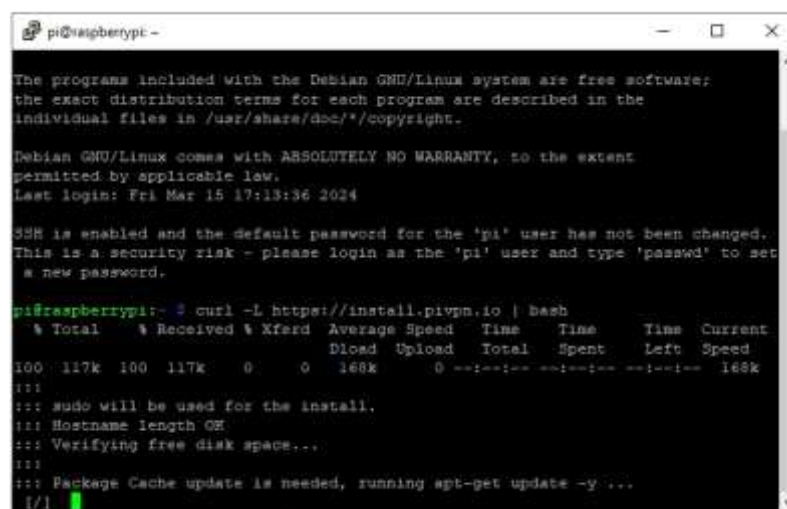
Причини вибору технології OpenVPN:

- технологія OpenVPN використовує потужні алгоритми шифрування, такі як AES, що забезпечує високий рівень захисту даних;
- підтримка різних платформ, що дозволяє легко інтегрувати VPN у різні середовища;
- технологія OpenVPN є проектом з відкритим вихідним кодом, що дозволяє спільноті виявляти та виправляти вразливості;

Для встановлення PiVPN необхідно виконати наступну команду.

```
curl -L https://install.pivpn.io | bash
```

Ця команда завантажує і запускає скрипт установки PiVPN (рис. 3.1).



```
pi@raspberrypi ~$ curl -L https://install.pivpn.io | bash
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Mar 15 17:13:36 2024

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~$ curl -L https://install.pivpn.io | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 117k  100 117k    0     0  168k      0 --:--:-- --:--:-- --:--:-- 168k
:::
::: sudo will be used for the install.
::: Hostname length OK
::: Verifying free disk space...
:::
::: Package Cache update is needed, running apt-get update -y ...
[!]
```

Рисунок 3.1 – Запуск скрипту на Raspberry PI

Після встановлення скрипт PiVPN проведе користувача через процес налаштування.

Для налаштування VPN серверу була обрана технологія OpenVPN, яка забезпечує високий рівень безпеки, гнучкість та підтримку різних платформ. OpenVPN використовує потужні алгоритми шифрування та є відкритим проєктом, що дозволяє швидко реагувати на вразливості та забезпечувати надійний захист даних.

Протокол UDP був обраний (рис. 3.2) для OpenVPN через його переваги у швидкості та ефективності передачі даних. На відміну від TCP, UDP не потребує встановлення з'єднання і перевірки кожного пакета, що робить його більш підходящим для додатків, де важлива швидкість передачі, таких як потокове відео та ігри.



Рисунок 3.2 – Вибір протоколу з'єднання

Під час налаштування PiVPN був обраний DNS-провайдер Google (рис. 3.3). DNS-сервери Google відомі своєю надійністю та високою швидкістю відповіді, що забезпечує стабільне та швидке розв'язання доменних імен для користувачів VPN[21].

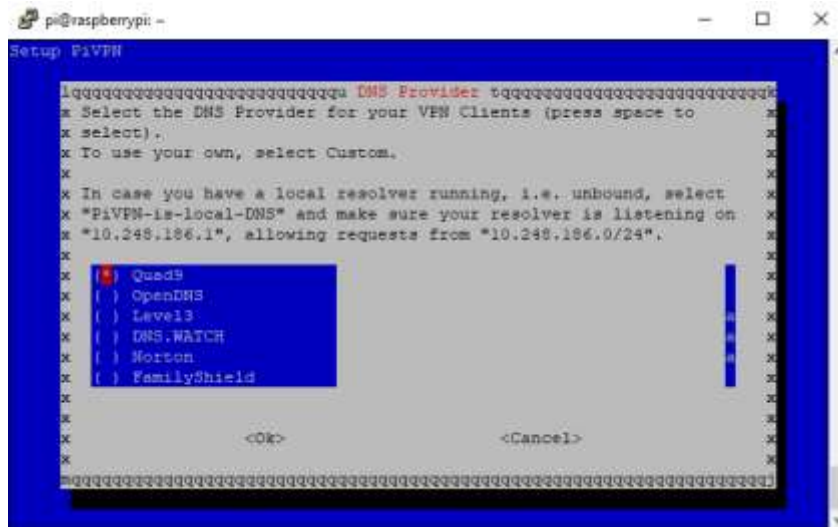


Рисунок 3.3 – Вибір DNS-провайдеру

Оскільки статична IP-адреса не доступна, для входу користувача використовується локальна IP-адреса RPI (рис. 3.4). Це забезпечує стабільний доступ до VPN серверу всередині локальної мережі. Використання локальної IP-адреси дозволяє уникнути проблем з динамічними IP-адресами, наданими провайдером інтернет-послуг. У конфігураційному файлі `raulch.ovpn` вказана локальна IP-адреса RPI для підключення користувача до VPN.

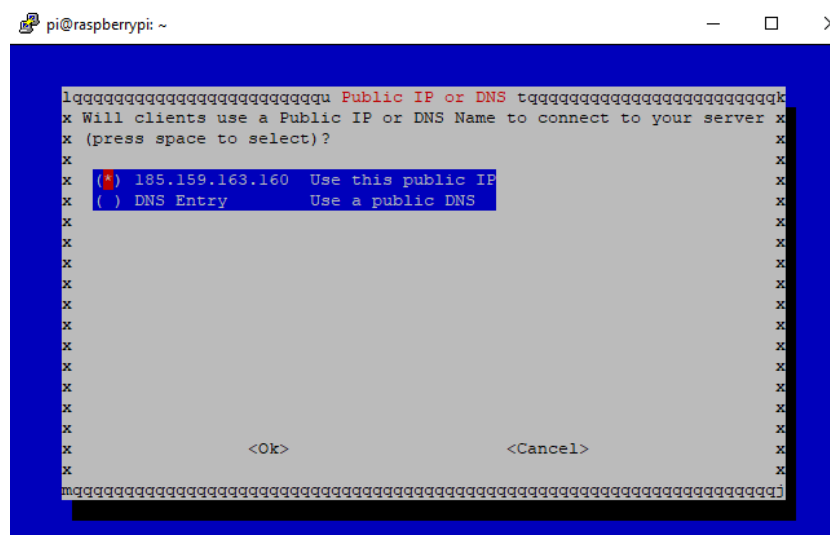


Рисунок 3.4 – Вибір підключення до серверу

Після завершення базової конфігурації згенеруємо сертифікати для клієнтів. Для цього виконаємо команду `pivpn add`.

Це дозволило створити клієнтський конфігураційний файл (рис. 3.5). У нашому випадку файл був названий paulch.ovpn.

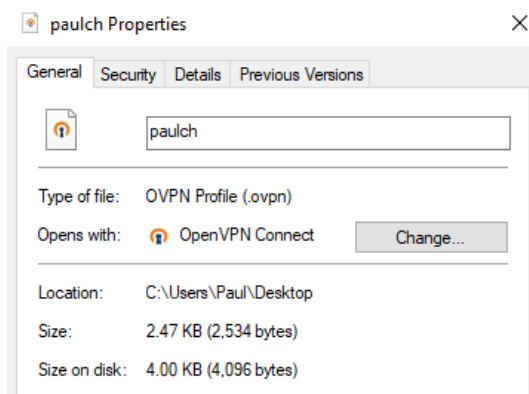


Рисунок 3.5 – Клієнтський конфігураційний файл

Після створення клієнтського файлу, його потрібно завантажити на пристрій клієнта і імпортувати у програму OpenVPN Connect.

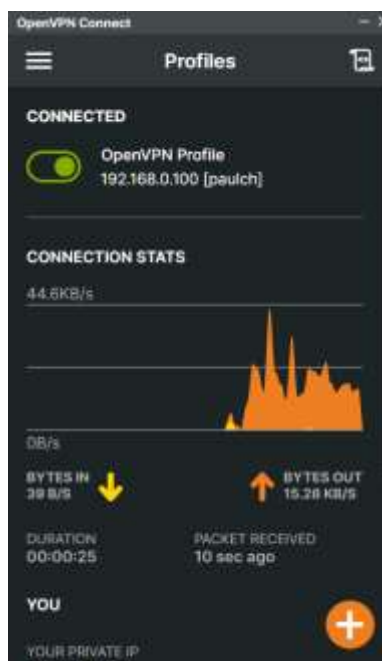


Рисунок 3.6 – Підключення користувача до VPN серверу

Таким чином, налаштування VPN серверу на RPI за допомогою PiVPN та OpenVPN забезпечило безпечний та надійний доступ до домашньої мережі та вебзастосунку HAD, навіть без наявності статичної IP-адреси (рис. 3.6).

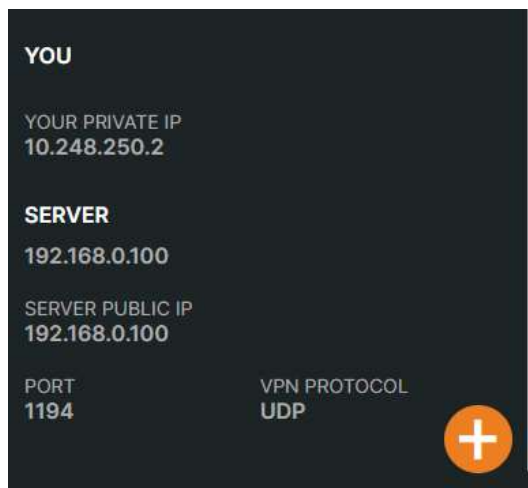


Рисунок 3.7 – Конфігурації підключеного користувача

Підключення починається з того, що користувач ініціює запит на з'єднання за допомогою OpenVPN клієнта. OpenVPN клієнт використовує конфігураційний файл для встановлення зашифрованого з'єднання з VPN сервером, що працює на RPI. Після автентифікації користувача VPN сервер відкриває тунель для шифрованого трафіку і перенаправляє цей трафік через мережу Інтернет (рис. 3.7).

3.2 Веб-застосунок Home Automation Dashboard

У цьому розділі детально розглядатиметься апаратне та програмне забезпечення, яке використовується для реалізації VPN сервера на базі RPI, а також проведемо експериментальні дослідження для оцінки його продуктивності та функціональності. Основна увага приділяється налаштуванню та роботі вебзастосунку HAD, який забезпечує управління розумним будинком через захищене VPN з'єднання.

Вебзастосунок HAD забезпечує користувачам зручний інтерфейс для керування пристроями розумного будинку через веб-браузер. Основні функції вебзастосунку включають:

- вебзастосунок дозволяє контролювати різні пристрої розумного будинку, такі як освітлення, термостати, камери спостереження тощо;
- користувачі можуть переглядати стан пристроїв у реальному часі;

Для розробки вебзастосунку HAD, було використано:

- *python* – мова програмування, що використовується для збору даних з сенсорів та пристроїв і передачі їх у Prometheus;
- *grafana* – платформа для аналітики та моніторингу, яка дозволяє створювати інтерактивні дашборди;
- *prometheus* – система для збору та зберігання метрик, яка забезпечує передачу даних у Grafana;

Цей вебзастосунок дозволить користувачам керувати пристроями розумного будинку через веб-інтерфейс[13].

Grafana надає інтуїтивно зрозумілий інтерфейс для створення дашбордів з різними типами візуалізацій, такими як графіки, таблиці, гейджі та інші. У нашій системі Grafana використовується для відображення даних, що збираються з різних сенсорів розумного дому.

Prometheus працює за принципом скрапінгу (scraping), тобто періодично опитує джерела даних (експортери) і зберігає отримані метрики. У нашій системі Prometheus використовується для зберігання даних, що збираються з сенсорів, і передачі їх до Grafana.

Python-скрипт збирає дані з різних сенсорів та пристроїв розумного дому і передає їх у Prometheus за допомогою бібліотеки `prometheus_client`[16].

3.3 Програмна реалізація

Для реалізації нашого застосунку нам необхідно імпортувати відповідні бібліотеки (рис. 3.8).

```
from prometheus_client import
start_http_server, Gauge
import requests
import time
```

Рисунок 3.8 – Імпорт потрібних бібліотек

– *prometheus_client* (*LIBRARY*) – бібліотека для роботи з Prometheus у Python;

– *requests* (*LIBRARY*) – бібліотека для виконання HTTP-запитів;

– *time* (*LIBRARY*) – стандартна бібліотека Python для роботи з часом;

Створемо метрики для збору даних з сенсорів і пристроїв.

```
temperature_gauge = Gauge('temperature', 'Temperature in different
rooms', ['sensor'])
humidity_gauge = Gauge('humidity', 'Humidity in different rooms',
['sensor'])
power_usage_gauge = Gauge('power_usage', 'Power usage of different
devices', ['device'])
co2_level_gauge = Gauge('co2_level', 'CO2 level in different rooms',
['sensor'])
noise_level_gauge = Gauge('noise_level', 'Noise level in different
rooms', ['sensor'])
```

Рисунок 3.9 – Створення метрик

Gauge - тип метрики, яка представляє одне числове значення, що може як збільшуватися, так і зменшуватися (рис. 3.9). Створені метрики включають показники температури, вологості, споживання енергії, рівня CO2 та рівня шуму.

Таблиця 3.1 – Реалізація функції для отримання даних з сенсорів через HTTP API.

Функція	Опис	URL для запиту
get_temperature(sensor)	Отримання температури з сенсора.	http://api.sensors.com/temperature?sensor={sensor}
get_humidity(sensor)	Отримання вологості з сенсора.	http://api.sensors.com/humidity?sensor={sensor}
get_power_usage(device)	Отримання даних про використання електроенергії з пристрою.	http://api.sensors.com/power_usage?device={device}
get_co2_level(sensor)	Отримання рівня CO2 з сенсора.	http://api.sensors.com/co2_level?sensor={sensor}
get_noise_level(sensor)	Отримання рівня шуму з сенсора.	http://api.sensors.com/noise_level?sensor={sensor}

Використовуємо бібліотеку requests для виконання HTTP-запитів до API сенсорів. Функції повертають значення метрик, отримані з відповідних сенсорів (табл. 3.1).

Функція для періодичного оновлення метрик з реальними даними.

```
def update_metrics():
    while True:
        try:

            temperature_gauge.labels(sensor="living_room").set(
                get_temperature("living_room"))
            humidity_gauge.labels(sensor="bedroom").set(get_humidity("bedroom"))
            power_usage_gauge.labels(device="heater").set(get_power_usage("heater"))
            power_usage_gauge.labels(device="air_conditioner").set(
                get_power_usage("air_conditioner"))
            power_usage_gauge.labels(device="washing_machine").set(
                get_power_usage("washing_machine"))
            co2_level_gauge.labels(sensor="kitchen").set(
                get_co2_level("kitchen"))
            noise_level_gauge.labels(sensor="living_room").set(
                get_noise_level("living_room"))
            noise_level_gauge.labels(sensor="bedroom").set(
                get_noise_level("bedroom"))
            noise_level_gauge.labels(sensor="kitchen").set(
                get_noise_level("kitchen"))
            noise_level_gauge.labels(sensor="bathroom").set(
                get_noise_level("bathroom"))
        except Exception as e:
            print(f"Error updating metrics: {e}")

        time.sleep(15)
```

Рисунок 3.10 – Функція оновлення метрик

Функція update_metrics періодично оновлює метрики, використовуючи дані, отримані з сенсорів (рис. 3.11). У разі виникнення помилки під час оновлення метрик, вона виводиться в консоль.

Запускаємо HTTP сервер для експорту метрик в Prometheus.

```
if __name__ == '__main__':  
    start_http_server(8000)  
    update_metrics() }
```

Рисунок 3.11 – Запуск HTTP сервер

`start_http_server(8000)` запускає HTTP сервер на порту 8000, через який Prometheus буде скрапити метрики (рис. 3.12). Функція `update_metrics` запускається для періодичного оновлення метрик.

Для відображення даних у Grafana створюються відповідні дашборди, які підключаються до джерела даних Prometheus. Дашборди можуть включати різні панелі для відображення температури, вологості, споживання енергії, рівня шуму, рівня CO2, статусу дверей та інших показників.

JSON-код конфігурації описує панель Grafana для відображення рівня CO2 на кухні. Панель використовує дані з Prometheus, налаштовані різні параметри для відображення даних та їх візуального представлення, такі як порогові значення, кольори та орієнтація. Панель налаштована на використання метрики `co2_level`, яку вона отримує з джерела даних Prometheus.

```
{  
  "datasource": {  
    "type": "prometheus",  
    "uid": "ado61yibj6br4d"  
  },  
}
```

Рисунок 3.12 – Джерело даних панелі CO2 level

Змінна `datasource` вказує джерело даних, яке використовується для цієї панелі (рис 3.13). `type` тип джерела даних, у цьому випадку `prometheus`. `uid` унікальний ідентифікатор джерела даних.

Змінна fieldConfig конфігурація для полів, що відображаються на панелі. defaults встановлює стандартні налаштування для всіх полів. mappings визначає мапінг значень. options параметри для мапінгу. match вказує умову для мапінгу, в даному випадку null. result результат для мапінгу, коли умова виконана.

```
"fieldConfig": {
  "defaults": {
    "mappings": [
      {
        "options": {
          "match": "null",
          "result": {
            "text": "N/A"
          }
        },
        "type": "special"
      }
    ],
    "thresholds": {
      "mode": "absolute",
      "steps": [
        {
          "color": "green",
          "value": null
        },
        {
          "color": "red",
          "value": 1000
        }
      ]
    },
    "color": {
      "mode": "thresholds"
    },
    "unit": "none"
  },
  "overrides": []
},
```

Рисунок 3.13 – Конфігурація полів панелі CO2 level

Змінна text відповідає за весь текст панелі, що відображається, коли значення null. type тип мапінгу, в даному випадку special. thresholds

встановлює порогові значення для кольорових індикацій. mode режим порогів, у цьому випадку absolute. steps порогові значення та кольори. color колір для значення. value значення порогу. color налаштування кольору. mode режим кольору, в даному випадку thresholds. unit одиниця вимірювання, у даному випадку none. overrides додаткові налаштування для специфічних полів, наразі порожній масив.

```
"gridPos": {  
  "h": 8,  
  "w": 12,  
  "x": 0,  
  "y": 1  
},
```

Рисунок 3.14 – Позичіонування панелі CO2 level

Змінна gridPos відповідає за позиціонування панелі на сітці дашборду (рис. 3.15). Висоту та ширину панелі регулює змінні h та w. Позиція по горизонталі та вертикалі регулює змінні x та y.

```
"id": 21,  
"maxDataPoints": 100,  
"options": {  
  "reduceOptions": {  
    "values": false,  
    "calcs": [  
      "lastNotNull"  
    ],  
    "fields": ""  
  },  
  "orientation": "auto",  
  "textMode": "auto",  
  "wideLayout": true,  
  "colorMode": "value",  
  "graphMode": "none",  
  "justifyMode": "auto",  
  "showPercentChange": false  
},  
"pluginVersion": "11.0.0",
```

Рисунок 3.15 – Додаткові налаштування панелі CO2 level

За ідентифікатор панелі відповідає змінна `id`. `maxDataPoints` – максимальна кількість точок даних, що відображаються на панелі. Додаткові параметри панелі `options`. `reduceOptions` налаштування агрегації даних. Змінна `values` визначає, чи використовувати всі значення. Вибір методів обчислення `calcs`, у даному випадку `lastNotNull` (останнє ненульове значення). Поля, до яких застосовується агрегація `fields`. `orientation` – орієнтація відображення, автоматична. `textMode` відповідає за режим відображення тексту, автоматичний. `colorMode` відповідає за режим кольору, у даному випадку `value` (відповідно до значення). `graphMode` – режим графіку. Режим вирівнювання `justifyMode`, автоматичний. Змінна `showPercentChange` відображає відсоткові зміни. `pluginVersion` версія плагіну, що використовується для панелі.

```
"targets": [  
  {  
    "datasource": {  
      "type": "prometheus",  
      "uid": "ado61yibj6br4d"  
    },  
    "editorMode": "code",  
    "expr": "co2_level",  
    "legendFormat": "__auto",  
    "range": true,  
    "refId": "A"  
  }  
],
```

Рисунок 3.16 – Налаштування передачі даних до панелі CO2 level

Змінна `targets` відповідає за джерела даних для панелі (рис. 3.17). Включає в себе `datasource`, який вказує джерело даних. `datasource` має дві змінні:

- `type` тип джерела даних, `prometheus`;
- `uid` унікальний ідентифікатор джерела даних;

`editorMode` режим редактора, `code`. `expr` це вираз Prometheus для отримання даних, у даному випадку `co2_level`.

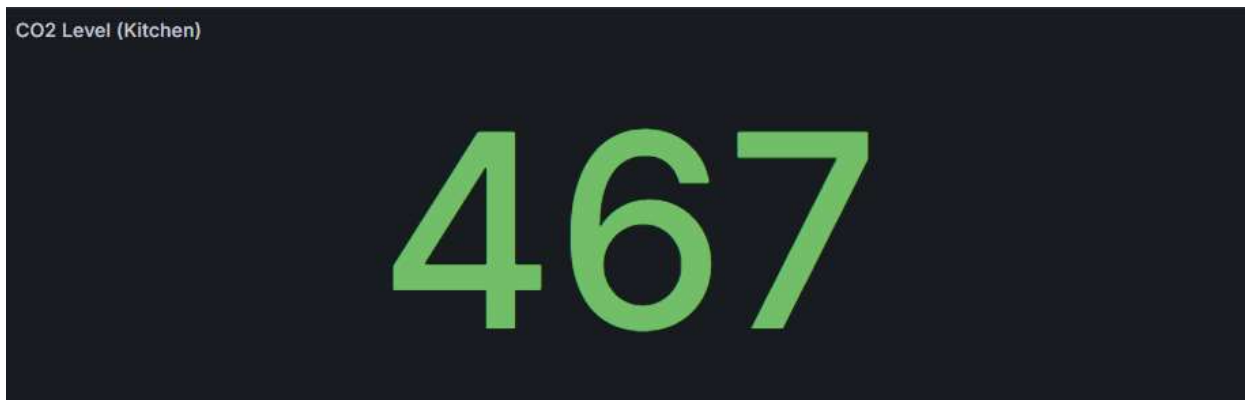


Рисунок 3.17 – Панель рівень кисню

```
{  
  "datasource": {  
    "type": "prometheus",  
    "uid": "ado61yibj6br4d"  
  },  
}
```

Рисунок 3.18 – Налаштування джерела даних панелі Power usage

Код відповідає за налаштування джерела даних (рис. 3.19). Вказується, що тип джерела даних - це Prometheus, а унікальний ідентифікатор джерела даних - "ado61yibj6br4d". Це означає, що панель буде використовувати дані з Prometheus для відображення метрик.

Розділ `fieldConfig` конфігурує відображення полів на панелі Power usage. `fieldConfig` описується чотирма змінними:

- змінна `defaults` відповідає за налаштування полів;
- змінна `thresholds` відповідає за порогові значення;
- змінна `steps` відповідає за опис налаштувань для порогових значень;
- змінна `color` задає колір значень на панелі;

```
"fieldConfig": {
  "defaults": {
    "mappings": [],
    "thresholds": {
      "mode": "absolute",
      "steps": [
        {
          "color": "red",
          "value": null
        },
        {
          "color": "green",
          "value": 20
        }
      ]
    },
    "color": {
      "mode": "thresholds"
    },
    "max": 100,
    "min": 0
  },
  "overrides": []
},
```

Рисунок 3.19 – Налаштування конфігурації полів панелі Power usage

В розділі "defaults" визначаються стандартні налаштування для полів. Мапінги відсутні, але встановлені порогові значення (thresholds). Пороговий режим заданий як "absolute", і він має два кроки: червоний колір для значень нижче 20 і зелений колір для значень від 20 і вище. Кольоровий режим визначається як "thresholds". Крім того, встановлюються максимальні (100) та мінімальні (0) значення. Розділ "overrides" порожній, що означає відсутність додаткових налаштувань для конкретних полів.

```
"gridPos": {
  "h": 8,
  "w": 12,
  "x": 12,
  "y": 1
},
```

Рисунок 3.20 – Позиціонування панелі Power usage на сітці дашборду

Цей код відповідає за позиціонування панелі на сітці дашборду. Вказується, що висота панелі (h) складає 8 одиниць, ширина (w) - 12 одиниць, а розташування по горизонталі (x) та вертикалі (y) визначаються як 12 та 1 відповідно. Це дозволяє розмістити панель на потрібному місці на дашборді.

```
"id": 23,  
  "options": {  
    "reduceOptions": {  
      "values": false,  
      "calcs": [  
        "mean"  
      ],  
      "fields": ""  
    },  
    "orientation": "auto",  
    "showThresholdLabels": false,  
    "showThresholdMarkers": true,  
    "sizing": "auto",  
    "minVizWidth": 75,  
    "minVizHeight": 75  
  },  
  "pluginVersion": "11.0.0",
```

Рисунок 3.21 – Основні параметри панелі Power usage

Код містить основні параметри панелі. Ідентифікатор панелі - 23. В параметрах "options" визначається, що агрегація даних (reduceOptions) не включає всі значення (values: false), використовує обчислення середнього значення (calcs: ["mean"]) та застосовується до всіх полів (fields: ""). Орієнтація відображення автоматична, мітки порогових значень не відображаються (showThresholdLabels: false), але маркери порогів відображаються (showThresholdMarkers: true). Розмір панелі автоматичний (sizing: "auto") з мінімальною шириною та висотою візуалізації 75 одиниць кожна. Версія плагіну, що використовується для цієї панелі, - "11.0.0".

```
"targets": [
  {
    "datasource": {
      "type": "prometheus",
      "uid": "ado61yibj6br4d"
    },
    "editorMode": "code",
    "expr": "power_usage",
    "interval": "",
    "legendFormat": "Power
usage",
    "range": true,
    "refId": "A"
  }
]
```

Рисунок 3.22 – Налаштування передачі цільових даних для панелі Power usage

Даний код описує цільові дані для панелі. Вказується, що тип джерела даних - це Prometheus з тим же унікальним ідентифікатором "ado61yibj6br4d". Режим редактора встановлений як "code", вираз Prometheus - "power_usage". Інтервал оновлення не визначений (interval: ""), формат легенди - "Power usage", а дані є діапазоном (range: true) з ідентифікатором посилання "A".

Визначаємо назву та тип панелі. Назва панелі - "Power Usage Heater (Wt)", а тип панелі - "gauge", що означає індикатор для відображення метрики.

Таким чином, цей JSON-код конфігурації описує панель Power usage, яка використовує дані з Prometheus для відображення споживання енергії обігрівачем у ватах за допомогою індикатора.

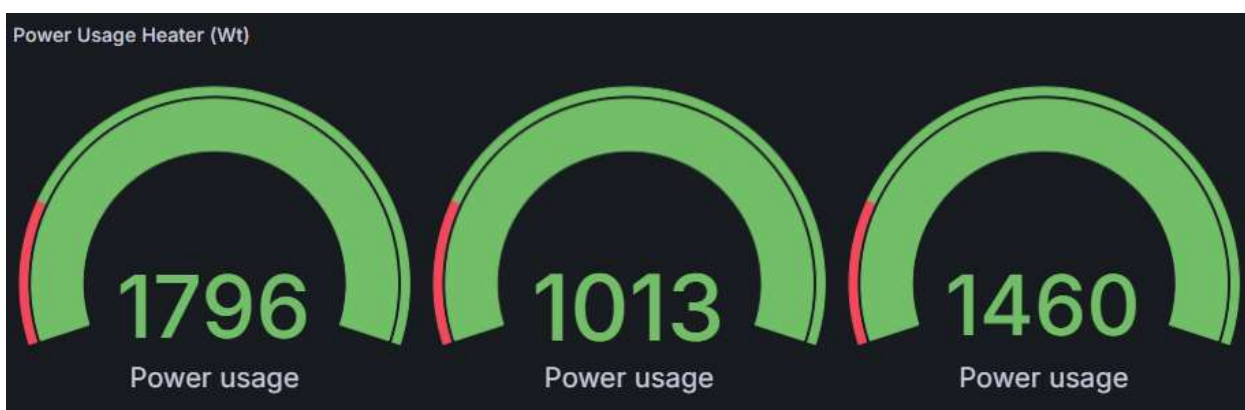


Рисунок 3.23 – Панель споживання енергії

Даний код описує конфігурацію полів панелі Temperature.

```
"fieldConfig": {
  "defaults": {
    "custom": {
      "drawStyle": "line",
      "lineInterpolation":
"linear",
      "barAlignment": 0,
      "lineWidth": 1,
      "fillOpacity": 10,
      "gradientMode": "none",
      "spanNulls": false,
      "insertNulls": false,
      "showPoints": "never",
      "pointSize": 5,
```

Рисунок 3.24(a) – Конфігурації полів панелі Temperature

У розділі "defaults" визначаються стандартні налаштування для полів. В підрозділі "custom" задаються параметри відображення: стиль лінії ("drawStyle": "line"), інтерполяція лінії ("lineInterpolation": "linear"), вирівнювання стовпців ("barAlignment": 0), ширина лінії ("lineWidth": 1), непрозорість заповнення ("fillOpacity": 10), режим градієнта ("gradientMode": "none"), поведінка при відсутності значень ("spanNulls": false, "insertNulls": false), відображення точок ("showPoints": "never"), розмір точок ("pointSize": 5).

```
"stacking": {  
  "mode": "none",  
  "group": "A"  
},  
  "axisPlacement": "auto",  
  "axisLabel": "",  
  "axisColorMode": "text",  
  "axisBorderShow": false,  
  "scaleDistribution": {  
    "type": "linear"  
  },  
  "axisCenteredZero": false,  
  "hideFrom": {  
    "tooltip": false,  
    "viz": false,  
    "legend": false  
  },  
  "thresholdsStyle": {  
    "mode": "off"  
  }  
},
```

Рисунок 3.24(б) – Конфігурації полів панелі Temperature

Налаштування стекування ("stacking": {"mode": "none", "group": "A"}), розміщення осі ("axisPlacement": "auto"), мітка осі ("axisLabel": ""), режим кольору осі ("axisColorMode": "text"), відображення межі осі ("axisBorderShow": false), розподіл шкали ("scaleDistribution": {"type": "linear"}), центрована нульова вісь ("axisCenteredZero": false), приховування елементів ("hideFrom": {"tooltip": false, "viz": false, "legend": false}), стиль порогових значень ("thresholdsStyle": {"mode": "off"}).

```
"color": {
  "mode": "palette-classic"
},
"mappings": [],
"thresholds": {
  "mode": "absolute",
  "steps": [
    {
      "color": "green",
      "value": null
    },
    {
      "color": "red",
      "value": 80
    }
  ]
},
"unit": "short"
},
"overrides": []
```

Рисунок 3.25 – Налаштування кольорового режиму панелі Temperature

Далі визначаються кольоровий режим ("color": {"mode": "palette-classic"}), відсутність мапінгів ("mappings": []), порогові значення ("thresholds": {"mode": "absolute", "steps": [{"color": "green", "value": null}, {"color": "red", "value": 80}]}), та одиниця вимірювання ("unit": "short"). Розділ "overrides" порожній, що означає відсутність додаткових налаштувань для конкретних полів.

```
"id": 25,
"options": {
  "tooltip": {
    "mode": "multi",
    "sort": "none",
    "maxHeight": 600
  },
  "legend": {
    "showLegend": true,
    "displayMode": "list",
    "placement": "bottom",
    "calcs": []
  }
},
```

Рисунок 3.26 – Основні параметри панелі Temperature

Даний код містить основні параметри панелі. Ідентифікатор панелі - 25. В параметрах "options" визначається режим підказок (tooltip: {"mode": "multi", "sort": "none", "maxHeight": 600}) та налаштування легенди (legend: {"showLegend": true, "displayMode": "list", "placement": "bottom", "calcs": []}).

```
"repeat": "deviceName",  
  "repeatDirection": "h",
```

Рисунок 3.27 – Повторення панелі Temperature для кожного пристрою

Визначаємо повторення панелі для кожного пристрою з назвою "deviceName" у горизонтальному напрямку ("repeatDirection": "h"). Це дозволяє створити декілька панелей для кожного пристрою з цією метрикою.

```
"targets": [  
  {  
    "datasource": {  
      "type": "prometheus",  
      "uid": "ado61yibj6br4d"  
    },  
    "editorMode": "code",  
    "expr": "temperature",  
    "legendFormat": "Temperature",  
    "range": true,  
    "refId": "A"  
  },  
  {  
    "datasource": {  
      "type": "prometheus",  
      "uid": "ado61yibj6br4d"  
    },  
    "editorMode": "code",  
    "expr": "humidity",  
    "legendFormat": "Humidity",  
    "range": true,  
    "refId": "B"  
  }  
],
```

Рисунок 3.28 – Налаштування передачі цільових даних панелі Temperature

У цьому коді описується цільові дані для панелі. Вказується, що тип джерела даних - Prometheus з тим же унікальним ідентифікатором "adob1yibj6br4d". Режим редактора встановлений як "code". Перший вираз Prometheus - "temperature" з форматом легенди "Temperature" та ідентифікатором посилання "A". Другий вираз Prometheus - "humidity" з форматом легенди "Humidity" та ідентифікатором посилання "B". Обидва вирази мають параметр "range": true, що вказує на використання діапазону значень.

Таким чином, цей JSON-код конфігурації описує панель Temperature, яка використовує дані з Prometheus для відображення температури та вологості у вигляді часових рядів.

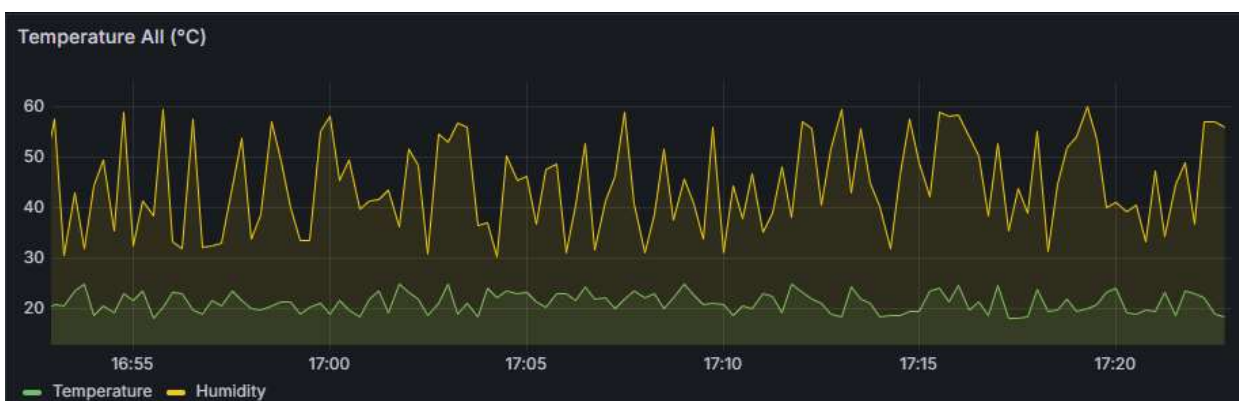


Рисунок 3.29 – Панель температури та вологості

Цей JSON-код описує панель Grafana для відображення рівня шуму в різних кімнатах розумного будинку. Панель використовує дані з Prometheus.

```

"fieldConfig": {
  "defaults": {
    "mappings": [],
    "thresholds": {
      "mode": "absolute",
      "steps": [
        {
          "color": "green",
          "value": null
        },
        {
          "color": "#EAB839",
          "value": 50
        },
        {
          "color": "red",
          "value": 70
        }
      ]
    },
    "color": {
      "mode": "thresholds"
    }
  },
  "overrides": []
},

```

Рисунок 3.30 – Конфігурація полів у панелі Noise level

Даний код налаштовує конфігурацію полів у панелі. У розділі "defaults" визначаються стандартні налаштування для полів, такі як порогові значення. Порогові значення налаштовуються у режимі "absolute" з трьома кроками: зелений колір для значень нижче 50, жовтий колір для значень від 50 до 70 і червоний колір для значень вище 70. Колірний режим встановлений на "thresholds". Розділ "overrides" порожній, що означає відсутність додаткових налаштувань для конкретних полів.

Даний код визначає ціль для панелі. Кожна ціль представляє метрику, яка буде відображатися на панелі. Всі вони використовують одне і те ж джерело даних Prometheus з ідентифікатором "adob1yibj6br4d" і режимом редактора "builder".

```
"targets": [  
  {  
    "datasource": {  
      "type": "prometheus",  
      "uid": "ado61yibj6br4d"  
    },  
    "disableTextWrap": false,  
    "editorMode": "builder",  
    "expr":  
"noise_level{sensor=\"bathroom\"}",  
    "fullMetaSearch": false,  
    "includeNullMetadata": true,  
    "instant": false,  
    "legendFormat": "Bathroom",  
    "range": true,  
    "refId": "A",  
    "useBackend": false  
  },
```

Рисунок 3.31 – Налаштування метрики у ванній кімнаті панелі Noise level

Метрика рівня шуму у ванній кімнаті з виразом
"noise_level{sensor="bathroom"}".

```
"datasource": {  
  "type": "prometheus",  
  "uid": "ado61yibj6br4d"  
},  
"disableTextWrap": false,  
"editorMode": "builder",  
"expr":  
"noise_level{sensor=\"bedroom\"}",  
"fullMetaSearch": false,  
"hide": false,  
"includeNullMetadata": true,  
"instant": false,  
"legendFormat": "Bedroom",  
"range": true,  
"refId": "B",  
"useBackend": false  
},
```

Рисунок 3.32 – Налаштування метрики у спальній кімнаті панелі Noise level

Код рівня шуму у спальній кімнаті з виразом
"noise_level{sensor="bedroom"}".

```
"datasource": {  
  "type": "prometheus",  
  "uid": "ado61yibj6br4d"  
},  
"disableTextWrap": false,  
"editorMode": "builder",  
"expr":  
"noise_level{sensor=\"kitchen\"}",  
"fullMetaSearch": false,  
"hide": false,  
"includeNullMetadata": true,  
"instant": false,  
"legendFormat": "Kitchen",  
"range": true,  
"refId": "C",  
"useBackend": false  
},
```

Рисунок 3.33 – Налаштування метрики на кухні панелі Noise level

Код рівня шуму на кухні з виразом "noise_level{sensor=\"kitchen\"}".

```
"datasource": {  
  "type": "prometheus",  
  "uid": "ado61yibj6br4d"  
},  
"disableTextWrap": false,  
"editorMode": "builder",  
"expr":  
"noise_level{sensor=\"living_room\"}",  
"fullMetaSearch": false,  
"hide": false,  
"includeNullMetadata": true,  
"instant": false,  
"legendFormat": "Living room",  
"range": true,  
"refId": "D",  
"useBackend": false  
}
```

Рисунок 3.34 – Налаштування метрики в гостинній панелі Noise level

Код рівня шуму в гостинній з виразом
"noise_level{sensor=\"living_room\"}".

Для кожної метрики встановлений формат легенди ("legendFormat") відповідно до назви кімнати, параметр "range" встановлений як true, що вказує на використання діапазону значень, та ідентифікатор посилання (refId) - "A", "B", "C" та "D" для кожної метрики відповідно.

Таким чином, цей JSON-код конфігурації описує панель Noise level, яка використовує дані з Prometheus для відображення рівня шуму в різних кімнатах будинку.

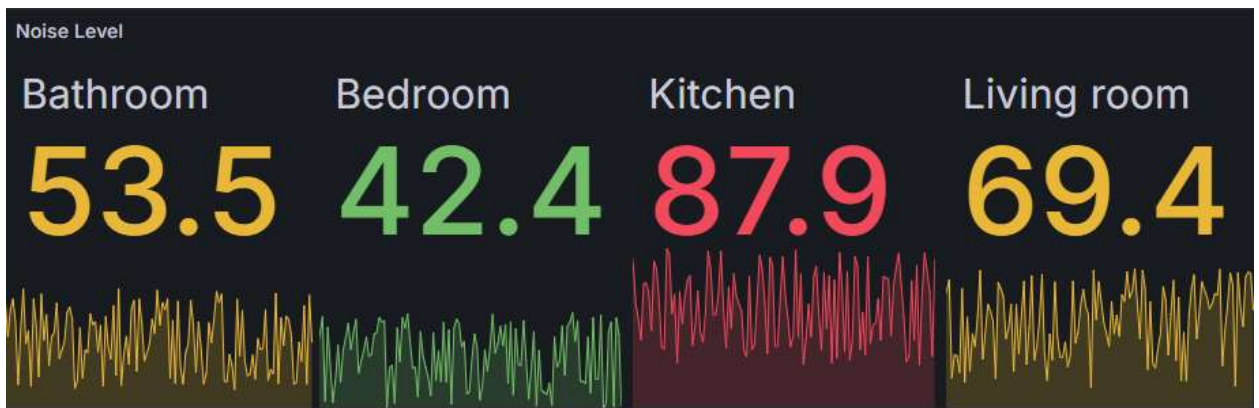


Рисунок 3.35 – Панель рівня шуму

Нижче наведена ілюстрація, яка демонструє повний дашборд з усіма панелями, що відображають дані про температуру, вологість, споживання електроенергії, рівень CO2 та рівень шуму в різних кімнатах розумного будинку.



Рисунок 3.36 – Вебзастосунок Home Automation Dashboard

Висновки до розділу 3

В ході роботи над системою розумного будинку було успішно реалізовано налаштування VPN-сервера на базі RPI, що забезпечило безпечний доступ до домашньої мережі з будь-якої точки світу. Це значно підвищує рівень безпеки системи, дозволяючи контролювати та керувати розумним будинком дистанційно.

Розроблений вебзастосунок HAD надає зручний інтерфейс для моніторингу та аналізу показників роботи системи в реальному часі. Використання Prometheus для збирання та зберігання метрик, а також Grafana для їх візуалізації, дозволяє ефективно відслідковувати зміни у стані різних сенсорів та пристроїв в будинку. Застосування мови програмування Python для інтеграції з сенсорами забезпечує гнучкість та масштабованість системи, дозволяючи легко додавати нові сенсори та пристрої.

Експериментальні дослідження показали, що розроблена система є стабільною та надійною, а також легко налаштовується під потреби користувача. Впровадження даної системи у домашнє середовище дозволяє не тільки підвищити комфорт проживання, але й значно знизити витрати на енергоспоживання за рахунок оптимізації роботи електроприладів.

Таким чином, виконана робота демонструє можливості сучасних технологій для створення ефективних та безпечних систем розумного будинку, що можуть бути інтегровані в повсякденне життя для підвищення його якості та безпеки.

ВИСНОВКИ

У ході роботи було досліджено та реалізовано VPN сервер на базі мікрокомп'ютера RPI, що забезпечує безпечний доступ до локальної мережі через Інтернет. OpenVPN забезпечив стабільну роботу та високу сумісність з різними пристроями та операційними системами, тоді як Wireguard продемонстрував вражаючу швидкість та ефективність завдяки своїй сучасній криптографічній базі та простоті налаштувань.

Додатково, у рамках роботи було розроблено веб-застосунок для автоматизації будинку, який дозволяє користувачам зручно керувати пристроями розумного будинку через VPN-з'єднання. Використання сервісу Grafana дало можливість створити інтерактивний та інтуїтивно зрозумілий інтерфейс для користувачів.

Результати проведених експериментальних досліджень показали, що VPN сервер на базі RPI здатний забезпечити стабільне та безпечне з'єднання для кількох одночасних користувачів, що підтверджує його життєздатність для реальних сценаріїв використання. Аналіз продуктивності та масштабованості системи дозволив виявити оптимальні налаштування для забезпечення максимальної ефективності роботи сервера в різних умовах.

На основі проведеної роботи можна зробити висновок, що RPI є надійною та економічно ефективною платформою для створення VPN серверів, які можуть бути використані як в особистих, так і в корпоративних цілях. Розроблений веб-застосунок для автоматизації будинку демонструє практичне застосування таких серверів та відкриває нові можливості для розвитку технологій розумних будинків. Подальші дослідження можуть бути спрямовані на оптимізацію продуктивності системи, інтеграцію додаткових функцій та розширення можливостей управління пристроями розумного будинку, а також на впровадження нових криптографічних протоколів для підвищення рівня безпеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Офіційний веб-сайт Raspberry Pi Foundation. URL: <https://www.raspberrypi.org> (дата звернення: 11.05.2024).
2. Офіційний форум спільноти Raspberry Pi. URL: <https://forums.raspberrypi.com/viewforum.php?f=84> (дата звернення: 11.05.2024).
3. Performance analysis of energy efficient life time improvement hybrid wireless sensor networks protocols for environmental monitoring systems: пат. 202321005524 Індія: H04W, G06F, H04L. № и 2023; заяв. 27.01.2023; опубл. 17.02.2023 (date of access: 14.05.2024).
4. A method of designing virtual kra process for small and large scale private cloud based firewall network by reducing coast of hardware: пат. 202221022660 Індія: H04L, G11B, B01J, H01L. № и 2022; заяв. 18.04.2022; опубл. 06.05.2022 (date of access: 14.05.2024).
5. An energy efficient smart location tracker assistance for elderly with cognitive disorder: пат. 202241050370 Індія: A61B, C07D, C07K, H05K, B60W. № и 2022; заяв. 03.09.2022; опубл. 09.09.2022 (date of access: 14.05.2024).
6. Tackling Tech Projects. *Medium*. URL: <https://medium.com/@crashwire1/tackling-tech-projects-build-a-vpn-server-with-a-raspberry-pi4-2b7fb8336be7> (date of access: 14.05.2024).
7. BBC «Is privacy dead in an online world?» URL: <https://www.bbc.com/news/technology-41483723> (дата звернення: 14.05.2024).
8. Best Reviews «Can a VPN Provider See My Traffic and History?» URL: <https://vpn-services.bestreviews.net/faq/can-vpn-provider-see-traffic-history/> (date of access: 14.05.2024).
9. Gabe Turner and the Security.org Research Team URL: <https://www.security.org/resources/vpn-consumer-report-annual/> (date of access: 14.05.2024).

10. Ashford W. Many search engine users unaware of personal data collection. 2019. URL: <https://www.computerweekly.com/news/252464048/Many-search-engine-users-unaware-of-personal-data-collection> (date of access: 14.05.2024).
11. Roser M., Ritchie H., Ortiz-Ospina E. Internet. 2015. URL: <https://ourworldindata.org/internet> (date of access: 14.05.2024).
12. Chinedu F., Emmanuel E. The functions of a Virtual Private Network. 2020. URL: <https://www.entrepreneurbusinessblog.com/functions-virtual-private-network-vpn/> (date of access: 14.05.2024).
13. Raspberry Pi – «Securing your Raspberry Pi» URL: <https://www.raspberrypi.com/documentation/computers/configuration.html> (date of access: 14.05.2024).
14. Kirk J. NordVPN Says Server Compromised Due to Misconfiguration. 2019. URL: <https://www.bankinfosecurity.com/nordvpn-says-server-compromised-due-to-misconfiguration-a-13278> (date of access: 14.05.2024).
15. Baig A. Free VPNs are a Privacy Nightmare: Here's Why. 2018. URL: <https://securitytoday.com/articles/2018/09/26/free-vpns-are-a-privacy-nightmare-heres-why.aspx> (date of access: 14.05.2024).
16. O'Driscoll A. Does your VPN Keep Logs? 120 VPN Logging Policies Revealed. 2020. URL: <https://www.comparitech.com/vpn/vpn-logging-policies/> (date of access: 14.05.2024).
17. Greenfield D. What are VPN Tunnels and How do They Work. 2019. URL: <https://www.catonetworks.com/blog/what-are-vpn-tunnels-and-how-do-they-work/> (date of access: 14.05.2024).
18. Teo Y. Why Do Hackers Hack? 2019. URL: <https://www.horangi.com/blog/why-do-hackers-hack> (date of access: 14.05.2024).
19. DuckDuckGo. URL: <https://spreadprivacy.com/three-reasons-why-the-nothing-to-hide-argument-is-flawed/> (date of access: 14.05.2024).
20. The Fastoe Blog. URL: <https://www.fastoe.com/blog/use-your-raspberry-pi-as-a-vpn-server-part-1> (date of access: 03.06.2024).

21. Xilinx Inc. URL: https://xilinx.github.io/Vitis_Libraries/security/2019.2/guide_L1/internals/chacha20.html (date of access: 03.06.2024).
22. Ethernet Bridging. *OpenVPN*. URL: <https://openvpn.net/community-resources/ethernet-bridging/> (date of access: 04.06.2024).
23. WireGuard. URL: <https://www.wireguard.com/#source-code> (date of access: 04.06.2024).
24. Index : wireguard-windows. URL: <https://git.zx2c4.com/wireguard-windows/about/> (date of access: 10.06.2024).
25. Prometheus. Docs. URL: <https://prometheus.io/docs/> (date of access: 10.06.2024).
26. Grafana. Start the Grafana server. URL: <https://grafana.com/docs/grafana/latest/setup-grafana/start-restart-grafana/> (date of access: 10.06.2024).
27. Как настроить vpn на роутере tp-link tl-mr3420. *Medium*. URL: <https://medium.com/@eliasbraeu/как-настроить-vpn-на-роутере-tp-link-tl-mr3420-e8d3b862ecc3> (дата звернення: 10.06.2024).

ДОДАТОК А

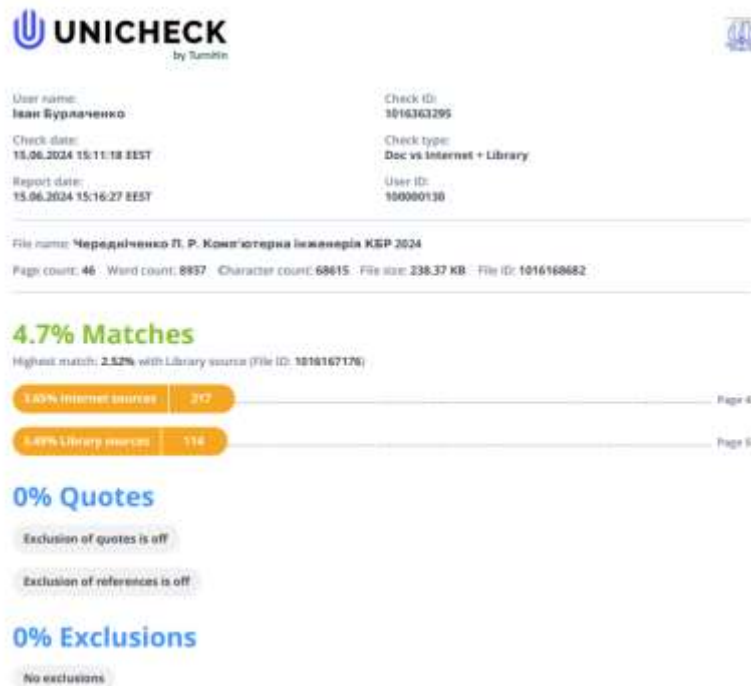
Звіт з антиплагіату

Довідка
про перевірку на унікальність пояснювальної записки
бакалаврської кваліфікаційної роботи на тему:
«VPN-сервер на базі Raspberry PI»

студента спеціальності 123 «Комп'ютерна інженерія», 405 групи
Чередніченко Павло Русланович

прізвище, ім'я, по-батькові

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck
Результат перевірки тексту кваліфікаційної бакалаврської роботи: схожість складає 4,7 %.



Здобувач:

студент 405 групи

_____ П. Р. Чередніченко

підпис

ініціали, прізвище

Дата: «__» _____ 2024 р.

Керівник:

ст. викладач

_____ І. С. Бурлаченко

підпис

ініціали, прізвище

ДОДАТОК Б

Код отримання даних з сенсорів

```
from prometheus_client import start_http_server, Gauge
import requests
import time

temperature_gauge = Gauge('temperature', 'Temperature in different
rooms', ['sensor'])
humidity_gauge = Gauge('humidity', 'Humidity in different rooms',
['sensor'])
power_usage_gauge = Gauge('power_usage', 'Power usage of different
devices', ['device'])
co2_level_gauge = Gauge('co2_level', 'CO2 level in different
rooms', ['sensor'])
noise_level_gauge = Gauge('noise_level', 'Noise level in different
rooms', ['sensor'])

def get_temperature(sensor):
    response =
requests.get(f'http://api.sensors.com/temperature?sensor={sensor}')
    return response.json()['value']

def get_humidity(sensor):
    response =
requests.get(f'http://api.sensors.com/humidity?sensor={sensor}')
    return response.json()['value']

def get_power_usage(device):
    response =
requests.get(f'http://api.sensors.com/power_usage?device={device}')
    return response.json()['value']

def get_co2_level(sensor):
```

```
        response =
requests.get(f'http://api.sensors.com/co2_level?sensor={sensor}')
        return response.json()['value']

    def get_noise_level(sensor):
        response =
requests.get(f'http://api.sensors.com/noise_level?sensor={sensor}')
        return response.json()['value']

    def update_metrics():
        while True:
            try:

temperature_gauge.labels(sensor="living_room").set(get_temperature("li
ving_room"))
humidity_gauge.labels(sensor="bedroom").set(get_humidity("bedroom"))
power_usage_gauge.labels(device="heater").set(get_power_usage("heater"
))
power_usage_gauge.labels(device="air_conditioner").set(get_power_usage
("air_conditioner"))
power_usage_gauge.labels(device="washing_machine").set(get_power_usage
("washing_machine"))
co2_level_gauge.labels(sensor="kitchen").set(get_co2_level("kitchen"))
noise_level_gauge.labels(sensor="living_room").set(get_noise_level("li
ving_room"))
noise_level_gauge.labels(sensor="bedroom").set(get_noise_level("bedroo
m"))
noise_level_gauge.labels(sensor="kitchen").set(get_noise_level("kitche
n"))
noise_level_gauge.labels(sensor="bathroom").set(get_noise_level("bathr
oom"))

            except Exception as e:
                print(f"Error updating metrics: {e}")
```

```
        time.sleep(15)

if __name__ == '__main__':

    start_http_server(8000)

    update_metrics()
```

ДОДАТОК В

Код конфігурації панелей дашбордів

```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": {
          "type": "datasource",
          "uid": "grafana"
        },
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "description": "A Dashboard for visualizing metrics regarding
smart home devices attached to your FRITZ!Box",
  "editable": true,
  "fiscalYearStartMonth": 0,
  "gnetId": 11345,
  "graphTooltip": 0,
  "id": 7,
  "links": [],
  "panels": [
    {
      "collapsed": false,
      "datasource": {
        "type": "prometheus",
        "uid": "ado61yibj6br4d"
      }
    }
  ]
}
```



```
},
"gridPos": {
  "h": 1,
  "w": 24,
  "x": 0,
  "y": 0
},
"id": 14,
"panels": [],
"targets": [
  {
    "datasource": {
      "type": "prometheus",
      "uid": "ado61yibj6br4d"
    },
    "refId": "A"
  }
],
"title": "Values",
"type": "row"
},
{
  "datasource": {
    "type": "prometheus",
    "uid": "ado61yibj6br4d"
  },
  "fieldConfig": {
    "defaults": {
      "color": {
        "mode": "thresholds"
      },
      "mappings": [
        {
          "options": {
```

```
    "match": "null",
    "result": {
      "text": "N/A"
    }
  },
  "type": "special"
}
],
"thresholds": {
  "mode": "absolute",
  "steps": [
    {
      "color": "green",
      "value": null
    },
    {
      "color": "red",
      "value": 1000
    }
  ]
},
"unit": "none"
},
"overrides": [],
},
"gridPos": {
  "h": 8,
  "w": 12,
  "x": 0,
  "y": 1
},
"id": 21,
"maxDataPoints": 100,
"options": {
```

```
"colorMode": "value",
"graphMode": "none",
"justifyMode": "auto",
"orientation": "auto",
"reduceOptions": {
  "calcs": [
    "lastNotNull"
  ],
  "fields": "",
  "values": false
},
"showPercentChange": false,
"textMode": "auto",
"wideLayout": true
},
"pluginVersion": "11.0.0",
"targets": [
  {
    "datasource": {
      "type": "prometheus",
      "uid": "ado61yibj6br4d"
    },
    "editorMode": "code",
    "expr": "co2_level",
    "legendFormat": "__auto",
    "range": true,
    "refId": "A"
  }
],
"title": "CO2 Level (Kitchen)",
"type": "stat"
},
{
  "datasource": {
```

```
"type": "prometheus",
"uid": "ado61yibj6br4d"
},
"fieldConfig": {
  "defaults": {
    "color": {
      "mode": "thresholds"
    },
    "mappings": [],
    "max": 100,
    "min": 0,
    "thresholds": {
      "mode": "absolute",
      "steps": [
        {
          "color": "red",
          "value": null
        },
        {
          "color": "green",
          "value": 20
        }
      ]
    }
  },
  "overrides": []
},
"gridPos": {
  "h": 8,
  "w": 12,
  "x": 12,
  "y": 1
},
"id": 23,
```

```
"options": {
  "minVizHeight": 75,
  "minVizWidth": 75,
  "orientation": "auto",
  "reduceOptions": {
    "calcs": [
      "mean"
    ],
    "fields": "",
    "values": false
  },
  "showThresholdLabels": false,
  "showThresholdMarkers": true,
  "sizing": "auto"
},
"pluginVersion": "11.0.0",
"targets": [
  {
    "datasource": {
      "type": "prometheus",
      "uid": "ado61yibj6br4d"
    },
    "editorMode": "code",
    "expr": "power_usage",
    "interval": "",
    "legendFormat": "Power usage",
    "range": true,
    "refId": "A"
  }
],
"title": "Power Usage Heater (Wt)",
"type": "gauge"
},
{
```

```
"collapsed": false,
"datasource": {
  "type": "prometheus",
  "uid": "ado61yibj6br4d"
},
"gridPos": {
  "h": 1,
  "w": 24,
  "x": 0,
  "y": 9
},
"id": 19,
"panels": [],
"targets": [
  {
    "datasource": {
      "type": "prometheus",
      "uid": "ado61yibj6br4d"
    },
    "refId": "A"
  }
],
"title": "Room metrics",
"type": "row"
},
{
  "datasource": {
    "type": "prometheus",
    "uid": "ado61yibj6br4d"
  },
  "fieldConfig": {
    "defaults": {
      "color": {
        "mode": "palette-classic"
```

```
},  
"custom": {  
  "axisBorderShow": false,  
  "axisCenteredZero": false,  
  "axisColorMode": "text",  
  "axisLabel": "",  
  "axisPlacement": "auto",  
  "barAlignment": 0,  
  "drawStyle": "line",  
  "fillOpacity": 10,  
  "gradientMode": "none",  
  "hideFrom": {  
    "legend": false,  
    "tooltip": false,  
    "viz": false  
  },  
  "insertNulls": false,  
  "lineInterpolation": "linear",  
  "lineWidth": 1,  
  "pointSize": 5,  
  "scaleDistribution": {  
    "type": "linear"  
  },  
  "showPoints": "never",  
  "spanNulls": false,  
  "stacking": {  
    "group": "A",  
    "mode": "none"  
  },  
  "thresholdsStyle": {  
    "mode": "off"  
  }  
},  
"mappings": [],
```

```
"thresholds": {
  "mode": "absolute",
  "steps": [
    {
      "color": "green",
      "value": null
    },
    {
      "color": "red",
      "value": 80
    }
  ]
},
"unit": "short"
},
"overrides": []
},
"gridPos": {
  "h": 8,
  "w": 12,
  "x": 0,
  "y": 10
},
"id": 25,
"options": {
  "legend": {
    "calcs": [],
    "displayMode": "list",
    "placement": "bottom",
    "showLegend": true
  },
  "tooltip": {
    "maxHeight": 600,
    "mode": "multi",
```



```
    "sort": "none"
  }
},
"repeat": "deviceName",
"repeatDirection": "h",
"targets": [
  {
    "datasource": {
      "type": "prometheus",
      "uid": "ado61yibj6br4d"
    },
    "editorMode": "code",
    "expr": "temperature",
    "legendFormat": "Temperature",
    "range": true,
    "refId": "A"
  },
  {
    "datasource": {
      "type": "prometheus",
      "uid": "ado61yibj6br4d"
    },
    "editorMode": "code",
    "expr": "humidity",
    "legendFormat": "Humidity",
    "range": true,
    "refId": "B"
  }
],
"title": "Temperature All (°C)",
"type": "timeseries"
},
{
  "datasource": {
```

```
"type": "prometheus",
"uid": "ado61yibj6br4d"
},
"fieldConfig": {
  "defaults": {
    "color": {
      "mode": "thresholds"
    },
    "mappings": [],
    "thresholds": {
      "mode": "absolute",
      "steps": [
        {
          "color": "green",
          "value": null
        },
        {
          "color": "#EAB839",
          "value": 50
        },
        {
          "color": "red",
          "value": 70
        }
      ]
    }
  },
  "overrides": []
},
"gridPos": {
  "h": 8,
  "w": 12,
  "x": 12,
  "y": 10
```

```
},  
"id": 26,  
"options": {  
  "colorMode": "value",  
  "graphMode": "area",  
  "justifyMode": "auto",  
  "orientation": "auto",  
  "reduceOptions": {  
    "calcs": [  
      "lastNotNull"  
    ],  
    "fields": "",  
    "values": false  
  },  
  "showPercentChange": false,  
  "textMode": "auto",  
  "wideLayout": true  
},  
"pluginVersion": "11.0.0",  
"targets": [  
  {  
    "datasource": {  
      "type": "prometheus",  
      "uid": "ado61yibj6br4d"  
    },  
    "disableTextWrap": false,  
    "editorMode": "builder",  
    "expr": "noise_level{sensor=\"bathroom\"}",  
    "fullMetaSearch": false,  
    "includeNullMetadata": true,  
    "instant": false,  
    "legendFormat": "Bathroom",  
    "range": true,  
    "refId": "A",
```

```
"useBackend": false
},
{
  "datasource": {
    "type": "prometheus",
    "uid": "ado61yibj6br4d"
  },
  "disableTextWrap": false,
  "editorMode": "builder",
  "expr": "noise_level{sensor=\"bedroom\"}",
  "fullMetaSearch": false,
  "hide": false,
  "includeNullMetadata": true,
  "instant": false,
  "legendFormat": "Bedroom",
  "range": true,
  "refId": "B",
  "useBackend": false
},
{
  "datasource": {
    "type": "prometheus",
    "uid": "ado61yibj6br4d"
  },
  "disableTextWrap": false,
  "editorMode": "builder",
  "expr": "noise_level{sensor=\"kitchen\"}",
  "fullMetaSearch": false,
  "hide": false,
  "includeNullMetadata": true,
  "instant": false,
  "legendFormat": "Kitchen",
  "range": true,
  "refId": "C",
```

```
    "useBackend": false
  },
  {
    "datasource": {
      "type": "prometheus",
      "uid": "ado61yibj6br4d"
    },
    "disableTextWrap": false,
    "editorMode": "builder",
    "expr": "noise_level{sensor=\"living_room\"}",
    "fullMetaSearch": false,
    "hide": false,
    "includeNullMetadata": true,
    "instant": false,
    "legendFormat": "Living room",
    "range": true,
    "refId": "D",
    "useBackend": false
  }
],
"title": "Noise Level",
"type": "stat"
}
],
"refresh": "30s",
"schemaVersion": 39,
"tags": [],
"templating": {
  "list": []
},
"time": {
  "from": "now-30m",
  "to": "now"
},
```

```
"timeRangeUpdatedDuringEditOrView": false,  
"timepicker": {  
  "refresh_intervals": [  
    "5s",  
    "10s",  
    "30s",  
    "1m",  
    "5m",  
    "15m",  
    "30m",  
    "1h",  
    "2h",  
    "1d"  
  ]  
},  
"timezone": "",  
"title": "Home Automation Dashboard",  
"uid": "ib8k12bZz",  
"version": 6,  
"weekStart": ""  
}
```